

# Information Flow Control via Dependency Tracking

Benedikt Nordhoff

November 24, 2021

## **Abstract**

We provide a characterisation of how information is propagated by program executions based on the tracking data and control dependencies within executions themselves. The characterisation might be used for deriving approximative safety properties to be targeted by static analyses or checked at runtime. We utilise a simple yet versatile control flow graph model as a program representation. As our model is not assumed to be finite it can be instantiated for a broad class of programs. The targeted security property is indistinguishable security where executions produce sequences of observations and only non-terminating executions are allowed to drop a tail of those.

A very crude approximation of our characterisation is slicing based on program dependence graphs, which we use as a minimal example and derive a corresponding soundness result.

For further details and applications refer to the authors upcoming dissertation.

# Contents

<b>1</b>	<b>Definitions</b>	<b>3</b>
1.1	Program Model . . . . .	3
1.1.1	Executions . . . . .	4
1.1.2	Well-formed Programs . . . . .	4
1.2	Security . . . . .	4
1.2.1	Observations . . . . .	4
1.2.2	Low equivalence of input states . . . . .	5
1.2.3	Termination . . . . .	5
1.2.4	Security Property . . . . .	5
1.3	Characterisation of Information Flows . . . . .	5
1.3.1	Post Dominance . . . . .	6
1.3.2	Control Dependence . . . . .	6
1.3.3	Control Slice . . . . .	6
1.3.4	Data Dependence . . . . .	7
1.3.5	Characterisation via Critical Paths . . . . .	7
1.3.6	Approximation via Single Critical Paths . . . . .	8
1.3.7	Further Definitions . . . . .	8
<b>2</b>	<b>Proofs</b>	<b>9</b>
2.1	Miscellaneous Facts . . . . .	9
2.2	Facts about Paths . . . . .	13
2.3	Facts about Post Dominators . . . . .	16
2.4	Facts about Control Dependencies . . . . .	23
2.5	Facts about Control Slices . . . . .	38
2.6	Facts about Observations . . . . .	65
2.7	Facts about Data . . . . .	69
2.8	Facts about Contradicting Paths . . . . .	71
2.9	Facts about Critical Observable Paths . . . . .	77
2.10	Correctness of the Characterisation . . . . .	86
2.11	Correctness of the Single Path Approximation . . . . .	87
<b>3</b>	<b>Example: Program Dependence Graphs</b>	<b>90</b>

# 1 Definitions

This section contains all necessary definitions of this development. Section 1.1 contains the structural definition of our program model which includes the security specification as well as abstractions of control flow and data. Executions of our program model are defined in section 1.1.1. Additional well-formedness properties are defined in section 1.1.2. Our security property is defined in section 1.2. Our characterisation of how information is propagated by executions of our program model is defined in section 1.3.5, for which the correctness result can be found in section 2.10. Section 1.3.6 contains an additional approximation of this characterisation whose correctness result can be found in section 2.11.

```
theory IFC
  imports Main
begin
```

## 1.1 Program Model

Our program model contains all necessary components for the remaining development and consists of:

```
record ('n, 'var, 'val, 'obs) ifc-problem =
— A set of nodes representing program locations:
  nodes :: '<'n set>
— An initial node where all executions start:
  entry :: '<'n>
— A final node where executions can terminate:
  return :: '<'n>
— An abstraction of control flow in the form of an edge relation:
  edges :: '<('n × 'n) set>
— An abstraction of variables written at program locations:
  writes :: '<'n ⇒ 'var set>
— An abstraction of variables read at program locations:
  reads :: '<'n ⇒ 'var set>
— A set of variables containing the confidential information in the initial state:
  hvars :: '<'var set>
— A step function on location state pairs:
  step :: '<('n × ('var ⇒ 'val)) ⇒ ('n × ('var ⇒ 'val))>
— An attacker model producing observations based on the reached state at certain locations:
  att :: '<'n → (('var ⇒ 'val) ⇒ 'obs)>
```

We fix a program in the following in order to define the central concepts. The necessary well-formedness assumptions will be made in section 1.1.2.

```
locale IFC-def =
fixes prob :: '<('n, 'var, 'val, 'obs) ifc-problem>
begin
```

Some short hands to the components of the program which we will utilise exclusively in the following.

```
definition nodes where <nodes = ifc-problem.nodes prob>
definition entry where <entry = ifc-problem.entry prob>
definition return where <return = ifc-problem.return prob>
definition edges where <edges = ifc-problem.edges prob>
definition writes where <writes = ifc-problem.writes prob>
definition reads where <reads = ifc-problem.reads prob>
definition hvars where <hvars = ifc-problem.hvars prob>
definition step where <step = ifc-problem.step prob>
definition att where <att = ifc-problem.att prob>
```

The components of the step function for convenience.

**definition** *suc* **where**  $\langle \text{suc } n \ \sigma = \text{fst } (\text{step } (n, \sigma)) \rangle$

**definition** *sem* **where**  $\langle \text{sem } n \ \sigma = \text{snd } (\text{step } (n, \sigma)) \rangle$

**lemma** *step-suc-sem*:  $\langle \text{step } (n, \sigma) = (\text{suc } n \ \sigma, \text{sem } n \ \sigma) \rangle$  **unfolding** *suc-def sem-def* **by** *auto*

### 1.1.1 Executions

In order to define what it means for a program to be well-formed, we first require concepts of executions and program paths.

The sequence of nodes visited by the execution corresponding to an input state.

**definition** *path* **where**

$\langle \text{path } \sigma \ k = \text{fst } ((\text{step} \sim^k) (\text{entry}, \sigma)) \rangle$

The sequence of states visited by the execution corresponding to an input state.

**definition** *kth-state* ( $\langle \cdot \rangle$  [111,111] 110) **where**

$\langle \sigma^k = \text{snd } ((\text{step} \sim^k) (\text{entry}, \sigma)) \rangle$

A predicate asserting that a sequence of nodes is a valid program path according to the control flow graph.

**definition** *is-path* **where**

$\langle \text{is-path } \pi = (\forall n. (\pi \ n, \pi \ (\text{Suc } n)) \in \text{edges}) \rangle$

**end**

### 1.1.2 Well-formed Programs

The following assumptions define our notion of valid programs.

**locale** *IFC* = *IFC-def*  $\langle \text{prob} \rangle$  **for** *prob*::  $\langle ('n, 'var, 'val, 'out) \text{ ifc-problem} \rangle$  +

**assumes** *ret-is-node*[*simp, intro*]:  $\langle \text{return} \in \text{nodes} \rangle$

**and** *entry-is-node*[*simp, intro*]:  $\langle \text{entry} \in \text{nodes} \rangle$

**and** *writes*:  $\langle \bigwedge v \ n. (\exists \sigma. \sigma \ v \neq \text{sem } n \ \sigma \ v) \implies v \in \text{writes } n \rangle$

**and** *writes-return*:  $\langle \text{writes } \text{return} = \{\} \rangle$

**and** *uses-writes*:  $\langle \bigwedge n \ \sigma \ \sigma'. (\forall v \in \text{reads } n. \sigma \ v = \sigma' \ v) \implies \forall v \in \text{writes } n. \text{sem } n \ \sigma \ v = \text{sem } n \ \sigma' \ v \rangle$

**and** *uses-suc*:  $\langle \bigwedge n \ \sigma \ \sigma'. (\forall v \in \text{reads } n. \sigma \ v = \sigma' \ v) \implies \text{suc } n \ \sigma = \text{suc } n \ \sigma' \rangle$

**and** *uses-att*:  $\langle \bigwedge n \ f \ \sigma \ \sigma'. \text{att } n = \text{Some } f \implies (\forall v \in \text{reads } n. \sigma \ v = \sigma' \ v) \implies f \ \sigma = f \ \sigma' \rangle$

**and** *edges-complete*[*intro, simp*]:  $\langle \bigwedge m \ \sigma. m \in \text{nodes} \implies (m, \text{suc } m \ \sigma) \in \text{edges} \rangle$

**and** *edges-return*:  $\langle \bigwedge x. (\text{return}, x) \in \text{edges} \implies x = \text{return} \rangle$

**and** *edges-nodes*:  $\langle \text{edges} \subseteq \text{nodes} \times \text{nodes} \rangle$

**and** *reaching-ret*:  $\langle \bigwedge x. x \in \text{nodes} \implies \exists \pi \ n. \text{is-path } \pi \wedge \pi \ 0 = x \wedge \pi \ n = \text{return} \rangle$

## 1.2 Security

We define our notion of security, which corresponds to what Bohannon et al. [1] refer to as indistinguishable security. In order to do so we require notions of observations made by the attacker, termination and equivalence of input states.

**context** *IFC-def*

**begin**

### 1.2.1 Observations

The observation made at a given index within an execution.

**definition** *obsp* **where**

$\langle \text{obsp } \sigma \ k = (\text{case } \text{att}(\text{path } \sigma \ k) \text{ of } \text{Some } f \Rightarrow \text{Some } (f (\sigma^k)) \mid \text{None} \Rightarrow \text{None}) \rangle$

The indices within a path where an observation is made.

**definition** *obs-ids* ::  $\langle (\text{nat} \Rightarrow 'n) \Rightarrow \text{nat set} \rangle$  **where**  
 $\langle \text{obs-ids } \pi = \{k. \text{att } (\pi \ k) \neq \text{None}\} \rangle$

A predicate relating an observable index to the number of observations made before.

**definition** *is-kth-obs* ::  $\langle (\text{nat} \Rightarrow 'n) \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow \text{bool} \rangle$  **where**  
 $\langle \text{is-kth-obs } \pi \ k \ i = (\text{card } (\text{obs-ids } \pi \cap \{..<i\}) = k \wedge \text{att } (\pi \ i) \neq \text{None}) \rangle$

The final sequence of observations made for an execution.

**definition** *obs* **where**  
 $\langle \text{obs } \sigma \ k = (\text{if } (\exists i. \text{is-kth-obs } (\text{path } \sigma) \ k \ i) \text{ then } \text{obsp } \sigma \ (\text{THE } i. \text{is-kth-obs } (\text{path } \sigma) \ k \ i) \text{ else } \text{None}) \rangle$

Comparability of observations.

**definition** *obs-prefix* ::  $\langle (\text{nat} \Rightarrow 'obs \text{ option}) \Rightarrow (\text{nat} \Rightarrow 'obs \text{ option}) \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \lesssim \rangle$  50) **where**  
 $\langle a \lesssim b \equiv \forall i. a \ i \neq \text{None} \longrightarrow a \ i = b \ i \rangle$

**definition** *obs-comp* (**infix**  $\langle \approx \rangle$  50) **where**  
 $\langle a \approx b \equiv a \lesssim b \vee b \lesssim a \rangle$

### 1.2.2 Low equivalence of input states

**definition** *restrict* (**infix**  $\langle | \rangle$  100) **where**  
 $\langle f|U = (\lambda n. \text{if } n \in U \text{ then } f \ n \text{ else } \text{undefined}) \rangle$

Two input states are low equivalent if they coincide on the non high variables.

**definition** *loweq* (**infix**  $\langle =_L \rangle$  50)  
**where**  $\langle \sigma =_L \sigma' = (\sigma|(-\text{hvars}) = \sigma'|(-\text{hvars})) \rangle$

### 1.2.3 Termination

An execution terminates iff it reaches the terminal node at any point.

**definition** *terminates* **where**  
 $\langle \text{terminates } \sigma \equiv \exists i. \text{path } \sigma \ i = \text{return} \rangle$

### 1.2.4 Security Property

The fixed program is secure if and only if for all pairs of low equivalent inputs the observation sequences are comparable and if the execution for an input state terminates then the observation sequence is not missing any observations.

**definition** *secure* **where**  
 $\langle \text{secure} \equiv \forall \sigma \ \sigma'. \sigma =_L \sigma' \longrightarrow (\text{obs } \sigma \approx \text{obs } \sigma' \wedge (\text{terminates } \sigma \longrightarrow \text{obs } \sigma' \lesssim \text{obs } \sigma)) \rangle$

## 1.3 Characterisation of Information Flows

We now define our characterisation of information flows which tracks data and control dependencies within executions. To do so we first require some additional concepts.

### 1.3.1 Post Dominance

We utilise the post dominance relation in order to define control dependence.

The basic post dominance relation.

**definition** *is-pd* (**infix**  $\langle pd \rightarrow \rangle$  50) **where**

$\langle y pd \rightarrow x \longleftrightarrow x \in nodes \wedge (\forall \pi n. is-path \pi \wedge \pi (0::nat) = x \wedge \pi n = return \longrightarrow (\exists k \leq n. \pi k = y)) \rangle$

The immediate post dominance relation.

**definition** *is-ipd* (**infix**  $\langle ipd \rightarrow \rangle$  50) **where**

$\langle y ipd \rightarrow x \longleftrightarrow x \neq y \wedge y pd \rightarrow x \wedge (\forall z. z \neq x \wedge z pd \rightarrow x \longrightarrow z pd \rightarrow y) \rangle$

**definition** *ipd* **where**

$\langle ipd x = (THE y. y ipd \rightarrow x) \rangle$

The post dominance tree.

**definition** *pdt* **where**

$\langle pdt = \{(x,y). x \neq y \wedge y pd \rightarrow x\} \rangle$

### 1.3.2 Control Dependence

An index on an execution path is control dependent upon another if the path does not visit the immediate post domiator of the node reached by the smaller index.

**definition** *is-cdi* ( $\langle - cd \rightarrow - \rangle$  [51,51,51]50) **where**

$\langle i cd \rightarrow k \longleftrightarrow is-path \pi \wedge k < i \wedge \pi i \neq return \wedge (\forall j \in \{k..i\}. \pi j \neq ipd (\pi k)) \rangle$

The largest control dependency of an index is the immediate control dependency.

**definition** *is-icdi* ( $\langle - icd \rightarrow - \rangle$  [51,51,51]50) **where**

$\langle n icd \rightarrow n' \longleftrightarrow is-path \pi \wedge n cd \rightarrow n' \wedge (\forall m \in \{n' < .. < n\}. \neg n cd \rightarrow m) \rangle$

For the definition of the control slice, which we will define next, we require the uniqueness of the immediate control dependency.

**lemma** *icd-uniq*: **assumes**  $\langle m icd \rightarrow n \rangle \langle m icd \rightarrow n' \rangle$  **shows**  $\langle n = n' \rangle$

**proof** –

```

{
  fix n n' assume *:  $\langle m icd \rightarrow n \rangle \langle m icd \rightarrow n' \rangle \langle n < n' \rangle$ 
  have  $\langle n' < m \rangle$  using * unfolding is-icdi-def is-cdi-def by auto
  hence  $\langle \neg m cd \rightarrow n' \rangle$  using * unfolding is-icdi-def by auto
  with *(2) have  $\langle False \rangle$  unfolding is-icdi-def by auto
}
thus ?thesis using assms by (metis linorder-neqE-nat)

```

qed

### 1.3.3 Control Slice

We utilise the control slice, that is the sequence of nodes visited by the control dependencies of an index, to match indices between executions.

**function** *cs*::  $\langle (nat \Rightarrow 'n) \Rightarrow nat \Rightarrow 'n list \rangle$  ( $\langle cs \rightarrow \rangle$  [51,70] 71) **where**

$\langle cs \pi n = (if (\exists m. n icd \rightarrow m) then (cs \pi (THE m. n icd \rightarrow m)) @ [\pi n] else [\pi n]) \rangle$

**by** *pat-completeness auto*

**termination**  $\langle cs \rangle$  **proof**

**show**  $\langle wf (measure snd) \rangle$  **by** *simp*

**fix**  $\pi n$

**define**  $m$  **where**  $\langle m == (The (is-icdi\ n\ \pi)) \rangle$   
**assume**  $\langle Ex (is-icdi\ n\ \pi) \rangle$   
**hence**  $\langle n\ icd^{\pi} \rightarrow m \rangle$  **unfolding**  $m$ -**def** **by**  $(metis (full-types) icd-uniq theI')$   
**hence**  $\langle m < n \rangle$  **unfolding**  $is-icdi$ -**def**  $is-cdi$ -**def** **by**  $simp$   
**thus**  $\langle ((\pi, The (is-icdi\ n\ \pi)), \pi, n) \in measure\ snd \rangle$  **by**  $(metis\ in-measure\ m-def\ snd-conv)$   
**qed**

**inductive**  $cs$ -**less** (**infix**  $\langle \prec \rangle$  50) **where**  
 $\langle length\ xs < length\ ys \implies take\ (length\ xs)\ ys = xs \implies xs \prec ys \rangle$

**definition**  $cs$ -**select** (**infix**  $\langle \jmath \rangle$  50) **where**  
 $\langle \pi \jmath xs = (THE\ k.\ cs^{\pi}\ k = xs) \rangle$

### 1.3.4 Data Dependence

Data dependence is defined straight forward. An index is data dependent upon another, if the index reads a variable written by the earlier index and the variable in question has not been written by any index in between.

**definition**  $is-ddi$  ( $\langle -\ dd^{\cdot} \rightarrow - \rangle$  [51,51,51,51] 50) **where**  
 $\langle n\ dd^{\pi, v} \rightarrow m \iff is-path\ \pi \wedge m < n \wedge v \in reads\ (\pi\ n) \cap (writes\ (\pi\ m)) \wedge (\forall\ l \in \{m < .. < n\}.\ v \notin writes\ (\pi\ l)) \rangle$

### 1.3.5 Characterisation via Critical Paths

With the above we define the set of critical paths which as we will prove characterise the matching points in executions where diverging data is read.

**inductive-set**  $cp$  **where**

— Any pair of low equivalent input states and indices where a diverging high variable is first read is critical.

$\langle \llbracket \sigma =_L \sigma';$   
 $\quad cs^{path}\ \sigma\ n = cs^{path}\ \sigma'\ n';$   
 $\quad h \in reads(path\ \sigma\ n);$   
 $\quad (\sigma^n)\ h \neq (\sigma^{m'})\ h;$   
 $\quad \forall\ k < n.\ h \notin writes(path\ \sigma\ k);$   
 $\quad \forall\ k' < n'.\ h \notin writes(path\ \sigma'\ k')$   
 $\rrbracket \implies ((\sigma, n), (\sigma', n')) \in cp \rangle \mid$

— If from a pair of critical indices in two executions there exist data dependencies from both indices to a pair of matching indices where the variable diverges, the later pair of indices is critical.

$\langle \llbracket ((\sigma, k), (\sigma', k')) \in cp;$   
 $\quad n\ dd^{path}\ \sigma, v \rightarrow k;$   
 $\quad n'\ dd^{path}\ \sigma', v \rightarrow k';$   
 $\quad cs^{path}\ \sigma\ n = cs^{path}\ \sigma'\ n';$   
 $\quad (\sigma^n)\ v \neq (\sigma^{m'})\ v$   
 $\rrbracket \implies ((\sigma, n), (\sigma', n')) \in cp \rangle \mid$

— If from a pair of critical indices the executions take different branches and one of the critical indices is a control dependency of an index that is data dependency of a matched index where diverging data is read and the variable in question is not written by the other execution after the executions first reached matching indices again, then the later matching pair of indices is critical.

$\langle [((\sigma, k), (\sigma', k')) \in cp;$   
 $n \text{ dd}^{\text{path}} \sigma, v \rightarrow l;$   
 $l \text{ cd}^{\text{path}} \sigma \rightarrow k;$   
 $cs^{\text{path}} \sigma \ n = cs^{\text{path}} \sigma' \ n';$   
 $\text{path } \sigma \ (\text{Suc } k) \neq \text{path } \sigma' \ (\text{Suc } k');$   
 $(\sigma^n) \ v \neq (\sigma'^n) \ v;$   
 $\forall j' \in \{(LEAST \ i'. \ k' < i' \wedge (\exists i. \ cs^{\text{path}} \sigma \ i = cs^{\text{path}} \sigma' \ i'))..<n'\}. \ v \notin \text{writes} \ (\text{path } \sigma' \ j')\}$   
 $\rangle \implies ((\sigma, n), (\sigma', n')) \in cp \rangle \mid$

— The relation is symmetric.

$\langle [((\sigma, k), (\sigma', k')) \in cp] \implies ((\sigma', k'), (\sigma, k)) \in cp \rangle$

Based on the set of critical paths, the critical observable paths are those that either directly reach observable nodes or are diverging control dependencies of an observable index.

**inductive-set** *cop* **where**

$\langle [((\sigma, n), (\sigma', n')) \in cp;$   
 $\text{path } \sigma \ n \in \text{dom att}$   
 $\rangle \implies ((\sigma, n), (\sigma', n')) \in cop \rangle \mid$

$\langle [((\sigma, k), (\sigma', k')) \in cp;$   
 $n \text{ cd}^{\text{path}} \sigma \rightarrow k;$   
 $\text{path } \sigma \ (\text{Suc } k) \neq \text{path } \sigma' \ (\text{Suc } k');$   
 $\text{path } \sigma \ n \in \text{dom att}$   
 $\rangle \implies ((\sigma, n), (\sigma', k')) \in cop \rangle$

### 1.3.6 Approximation via Single Critical Paths

For applications we also define a single execution approximation.

**definition** *is-dcdi-via* ( $\langle \text{- dcd}^{\text{-}} \rightarrow \text{- via } \text{-} \rightarrow [51, 51, 51, 51, 51, 51] \ 50 \rangle$  **where**

$\langle n \text{ dcd}^{\pi, v} \rightarrow m \text{ via } \pi' \ m' = (\text{is-path } \pi \wedge m < n \wedge (\exists l' \ n'. \ cs^{\pi} \ m = cs^{\pi'} \ m' \wedge cs^{\pi} \ n = cs^{\pi'} \ n' \wedge n' \text{ dd}^{\pi', v} \rightarrow l' \wedge l' \text{ cd}^{\pi'} \rightarrow m')) \wedge (\forall l \in \{m..<n\}. \ v \notin \text{writes}(\pi \ l)) \rangle$

**inductive-set** *scp* **where**

$\langle [h \in \text{hvars}; h \in \text{reads} \ (\text{path } \sigma \ n); (\forall k < n. \ h \notin \text{writes}(\text{path } \sigma \ k))] \implies (\text{path } \sigma, n) \in scp \rangle \mid$   
 $\langle [(\pi, m) \in scp; n \text{ cd}^{\pi} \rightarrow m] \implies (\pi, n) \in scp \rangle \mid$   
 $\langle [(\pi, m) \in scp; n \text{ dd}^{\pi, v} \rightarrow m] \implies (\pi, n) \in scp \rangle \mid$   
 $\langle [(\pi, m) \in scp; (\pi', m') \in scp; n \text{ dcd}^{\pi, v} \rightarrow m \text{ via } \pi' \ m'] \implies (\pi, n) \in scp \rangle$

**inductive-set** *scop* **where**

$\langle [(\pi, n) \in scp; \pi \ n \in \text{dom att}] \implies (\pi, n) \in scop \rangle$

### 1.3.7 Further Definitions

The following concepts are utilised by the proofs.

**inductive** *contradicts* (**infix**  $\langle \text{c} \rangle \ 50$ ) **where**

$\langle [cs^{\pi'} \ k' \prec cs^{\pi} \ k; \pi = \text{path } \sigma; \pi' = \text{path } \sigma'; \pi \ (\text{Suc} \ (\pi \ cs^{\pi'} \ k')) \neq \pi' \ (\text{Suc} \ k')] \implies (\sigma', k') \ \text{c} \ (\sigma, k) \rangle \mid$   
 $\langle [cs^{\pi'} \ k' = cs^{\pi} \ k; \pi = \text{path } \sigma; \pi' = \text{path } \sigma'; \sigma^k \ \uparrow \ (\text{reads} \ (\pi \ k)) \neq \sigma'^{k'} \ \uparrow \ (\text{reads} \ (\pi \ k))] \implies (\sigma', k') \ \text{c} \ (\sigma, k) \rangle$

**definition** *path-shift* (**infixl**  $\langle \ll \rangle \ 51$ ) **where**

[simp]:  $\langle \pi \ll m = (\lambda \ n. \ \pi \ (m+n)) \rangle$

**definition** *path-append* ::  $\langle (\text{nat} \Rightarrow 'n) \Rightarrow \text{nat} \Rightarrow (\text{nat} \Rightarrow 'n) \Rightarrow (\text{nat} \Rightarrow 'n) \rangle$  ( $\langle \text{-} \ @ \text{-} \rightarrow [0, 0, 999] \ 51 \rangle$ ) **where**



[simp]:  $\langle \pi @^m \pi' = (\lambda n. (\text{if } n \leq m \text{ then } \pi \ n \text{ else } \pi' (n-m))) \rangle$

**definition** *eq-up-to* ::  $\langle (\text{nat} \Rightarrow 'n) \Rightarrow \text{nat} \Rightarrow (\text{nat} \Rightarrow 'n) \Rightarrow \text{bool} \rangle$  ( $\langle \_ = \_ \rightarrow [55,55,55] \ 50 \rangle$ ) **where**  
 $\langle \pi =_k \pi' = (\forall \ i \leq k. \ \pi \ i = \pi' \ i) \rangle$

end

## 2 Proofs

### 2.1 Miscellaneous Facts

**lemma** *option-neq-cases*: **assumes**  $\langle x \neq y \rangle$  **obtains** (*none1*) *a* **where**  $\langle x = \text{None} \rangle \langle y = \text{Some } a \rangle$  | (*none2*) *a* **where**  $\langle x = \text{Some } a \rangle \langle y = \text{None} \rangle$  | (*some*) *a b* **where**  $\langle x = \text{Some } a \rangle \langle y = \text{Some } b \rangle \langle a \neq b \rangle$  **using** *assms* **by** *fastforce*

**lemmas** *nat-sym-cases*[*case-names less sym eq*] = *linorder-less-wlog*

**lemma** *mod-bound-instance*: **assumes**  $\langle j < (i::\text{nat}) \rangle$  **obtains** *j'* **where**  $\langle k < j' \rangle$  **and**  $\langle j' \bmod i = j \rangle$  **proof** –  
**have**  $\langle k < \text{Suc } k * i + j \rangle$  **using** *assms less-imp-Suc-add* **by** *fastforce*  
**moreover**  
**have**  $\langle (\text{Suc } k * i + j) \bmod i = j \rangle$  **by** (*metis assms mod-less mod-mult-self3*)  
**ultimately show** *thesis* **using** *that* **by** *auto*  
**qed**

**lemma** *list-neq-prefix-cases*: **assumes**  $\langle ls \neq ls' \rangle$  **and**  $\langle ls \neq \text{Nil} \rangle$  **and**  $\langle ls' \neq \text{Nil} \rangle$   
**obtains** (*diverge*) *xs x' ys ys'* **where**  $\langle ls = xs @ [x] @ ys \rangle \langle ls' = xs @ [x'] @ ys' \rangle \langle x \neq x' \rangle$  |  
(*prefix1*) *xs* **where**  $\langle ls = ls' @ xs \rangle$  **and**  $\langle xs \neq \text{Nil} \rangle$  |  
(*prefix2*) *xs* **where**  $\langle ls @ xs = ls' \rangle$  **and**  $\langle xs \neq \text{Nil} \rangle$

**using** *assms* **proof** (*induct*  $\langle \text{length } ls \rangle$  *arbitrary*:  $\langle ls \rangle \langle ls' \rangle$  *rule*: *less-induct*)

**case** (*less ls ls'*)

**obtain** *z zs z' zs'* **where**

*lz*:  $\langle ls = z \# zs \rangle \langle ls' = z' \# zs' \rangle$  **by** (*metis list.exhaust less(6,7)*)

**show**  $\langle ?\text{case} \rangle$  **proof** *cases*

**assume** *zz*:  $\langle z = z' \rangle$

**hence** *zsz*:  $\langle zs \neq zs' \rangle$  **using** *less(5)* *lz* **by** *auto*

**have** *lenz*:  $\langle \text{length } zs < \text{length } ls \rangle$  **using** *lz* **by** *auto*

**show**  $\langle ?\text{case} \rangle$  **proof**(*cases*  $\langle zs = \text{Nil} \rangle$ )

**assume** *zs*:  $\langle zs = \text{Nil} \rangle$

**hence**  $\langle zs' \neq \text{Nil} \rangle$  **using** *zsz* **by** *auto*

**moreover**

**have**  $\langle ls @ zs' = ls' \rangle$  **using** *zs lz zz* **by** *auto*

**ultimately**

**show** *thesis* **using** *less(4)* **by** *blast*

**next**

**assume** *zs*:  $\langle zs \neq \text{Nil} \rangle$

**show** *thesis* **proof** (*cases*  $\langle zs' = \text{Nil} \rangle$ )

**assume**  $\langle zs' = \text{Nil} \rangle$

**hence**  $\langle ls = ls' @ zs \rangle$  **using** *lz zz* **by** *auto*

**thus** *thesis* **using** *zs less(3)* **by** *blast*

**next**

**assume** *zs'*:  $\langle zs' \neq \text{Nil} \rangle$

{ **fix** *xs x ys x' ys'*

**assume**  $\langle zs = xs @ [x] @ ys \rangle \langle zs' = xs @ [x'] @ ys' \rangle$  **and** *xx*:  $\langle x \neq x' \rangle$

**hence**  $\langle ls = (z \# xs) @ [x] @ ys \rangle \langle ls' = (z \# xs) @ [x'] @ ys' \rangle$  **using** *lz zz* **by** *auto*

**hence** *thesis* **using** *less(2)* *xx* **by** *blast*

} **note** \* = *this*

```

{ fix xs
  assume ⟨zs = zs' @ xs⟩ and xs: ⟨xs ≠ []⟩
  hence ⟨ls = ls' @ xs⟩ using lz zz by auto
  hence ⟨thesis⟩ using xs less(3) by blast
} note ** = this
{ fix xs
  assume ⟨zs@xs = zs'⟩ and xs: ⟨xs ≠ []⟩
  hence ⟨ls@xs = ls'⟩ using lz zz by auto
  hence ⟨thesis⟩ using xs less(4) by blast
} note *** = this
have ⟨(∧xs x ys x' ys'. zs = xs @ [x] @ ys ⇒ zs' = xs @ [x'] @ ys' ⇒ x ≠ x' ⇒ thesis) ⇒
  (∧xs. zs = zs' @ xs ⇒ xs ≠ [] ⇒ thesis) ⇒
  (∧xs. zs @ xs = zs' ⇒ xs ≠ [] ⇒ thesis) ⇒ thesis⟩
using less(1)[OF lenz - - zsz zs zs'] .
thus ⟨thesis⟩ using * ** *** by blast
qed
qed
next
  assume ⟨z ≠ z'⟩
  moreover
  have ⟨ls = []@[z]@zs⟩ ⟨ls' = []@[z']@zs'⟩ using lz by auto
  ultimately show ⟨thesis⟩ using less(2) by blast
qed
qed

lemma three-cases: assumes ⟨A ∨ B ∨ C⟩ obtains ⟨A⟩ | ⟨B⟩ | ⟨C⟩ using assms by auto

lemma insert-greater: ⟨∀ x ∈ set ls. x < y ⇒ insert y ls = ls@[y]⟩ by (induction ⟨ls⟩, auto)

lemma insert-append-first: assumes ⟨∀ y ∈ set ys. x ≤ y⟩ shows ⟨insert x (xs@ys) = insert x xs @ ys⟩ using
assms by (induction ⟨xs⟩, auto, metis insert-is-Cons)

lemma sorted-list-of-set-append: assumes ⟨finite xs⟩ ⟨finite ys⟩ ⟨∀ x ∈ xs. ∀ y ∈ ys. x < y⟩ shows ⟨sorted-list-of-set
(xs ∪ ys) = sorted-list-of-set xs @ (sorted-list-of-set ys)⟩
using assms(1,3) proof (induction ⟨xs⟩)
  case empty thus ⟨?case⟩ by simp
next
  case (insert x xs)
  hence iv: ⟨sorted-list-of-set (xs ∪ ys) = sorted-list-of-set xs @ sorted-list-of-set ys⟩ by blast
  have le: ⟨∀ y ∈ set (sorted-list-of-set ys). x < y⟩ using insert(4) assms(2) sorted-list-of-set by auto
  have ⟨sorted-list-of-set (insert x xs ∪ ys) = sorted-list-of-set (insert x (xs ∪ ys))⟩ by auto
  also
  have ⟨... = insert x (sorted-list-of-set (xs ∪ ys))⟩ by (metis Un-iff assms(2) finite-Un insert.hyps(1) in-
sert.hyps(2) insert.prem1 insertI1 less-irrefl sorted-list-of-set-insert)
  also
  have ⟨... = insert x (sorted-list-of-set xs @ sorted-list-of-set ys)⟩ using iv by simp
  also
  have ⟨... = insert x (sorted-list-of-set xs) @ sorted-list-of-set ys⟩ by (metis le insert-append-first less-le-not-le)
  also
  have ⟨... = sorted-list-of-set (insert x xs) @ sorted-list-of-set ys⟩ using sorted-list-of-set-insert[OF insert(1), of
⟨x⟩] insert(2) by auto
  finally
  show ⟨?case⟩ .
qed

lemma filter-insert: ⟨sorted xs ⇒ filter P (insert x xs) = (if P x then insert x (filter P xs) else filter P xs)⟩

```

by (induction  $\langle xs \rangle$ , simp) (metis filter-insort filter-insort-triv map-ident)

**lemma filter-sorted-list-of-set:** assumes  $\langle \text{finite } xs \rangle$  shows  $\langle \text{filter } P (\text{sorted-list-of-set } xs) = \text{sorted-list-of-set } \{x \in xs. P x\} \rangle$  using assms **proof**(induction  $\langle xs \rangle$ )

case empty thus  $\langle ?case \rangle$  by simp

next

case (insert  $x xs$ )

have \*:  $\langle \text{set } (\text{sorted-list-of-set } xs) = xs \rangle \langle \text{sorted } (\text{sorted-list-of-set } xs) \rangle \langle \text{distinct } (\text{sorted-list-of-set } xs) \rangle$  by (auto simp add: insert.hyps(1))

have \*\*:  $\langle P x \implies \{y \in \text{insert } x xs. P y\} = \text{insert } x \{y \in xs. P y\} \rangle$  by auto

have \*\*\*:  $\langle \neg P x \implies \{y \in \text{insert } x xs. P y\} = \{y \in xs. P y\} \rangle$  by auto

**note** filter-insort[OF \*(2),of  $\langle P \rangle \langle x \rangle$ ] sorted-list-of-set-insert[OF insert(1), of  $\langle x \rangle$ ] insert(2,3) \*\* \*\*\*

thus  $\langle ?case \rangle$  by (metis (mono-tags) \*(1) List.finite-set distinct-filter distinct-insort distinct-sorted-list-of-set set-filter sorted-list-of-set-insert)

qed

**lemma unbounded-nat-set-infinite:** assumes  $\langle \forall (i::\text{nat}). \exists j \geq i. j \in A \rangle$  shows  $\langle \neg \text{finite } A \rangle$  using assms by (metis finite-nat-set-iff-bounded-le not-less-eq-eq)

**lemma infinite-ascending:** assumes  $\text{nf}: \langle \neg \text{finite } (A::\text{nat set}) \rangle$  obtains  $f$  where  $\langle \text{range } f = A \rangle \langle \forall i. f i < f (\text{Suc } i) \rangle$  **proof**

let  $\langle ?f \rangle = \langle \lambda i. (\text{LEAST } a. a \in A \wedge \text{card } (A \cap \{..<a\}) = i) \rangle$

{ fix  $i$

obtain  $a$  where  $\langle a \in A \rangle \langle \text{card } (A \cap \{..<a\}) = i \rangle$

**proof** (induction  $\langle i \rangle$  arbitrary:  $\langle \text{thesis} \rangle$ )

case 0

let  $\langle ?a0 \rangle = \langle (\text{LEAST } a. a \in A) \rangle$

have  $\langle ?a0 \in A \rangle$  by (metis LeastI empty-iff finite.emptyI nf set-eq-iff)

moreover

have  $\langle \bigwedge b. b \in A \implies ?a0 \leq b \rangle$  by (metis Least-le)

hence  $\langle \text{card } (A \cap \{..<?a\}) = 0 \rangle$  by force

ultimately

show  $\langle ?case \rangle$  using 0 by blast

next

case (Suc  $i$ )

obtain  $a$  where  $\text{aa}: \langle a \in A \rangle$  and  $\text{card}: \langle \text{card } (A \cap \{..<a\}) = i \rangle$  using Suc.IH by metis

have  $\text{nf}': \langle \sim \text{finite } (A - \{..a\}) \rangle$  using nf by auto

let  $\langle ?b \rangle = \langle \text{LEAST } b. b \in A - \{..a\} \rangle$

have  $\text{bin}: \langle ?b \in A - \{..a\} \rangle$  by (metis LeastI empty-iff finite.emptyI nf' set-eq-iff)

have  $\text{le}: \langle \bigwedge c. c \in A - \{..a\} \implies ?b \leq c \rangle$  by (metis Least-le)

have  $\text{ab}: \langle a < ?b \rangle$  using bin by auto

have  $\langle \bigwedge c. c \in A \implies c < ?b \implies c \leq a \rangle$  using le by force

hence  $\langle A \cap \{..<?b\} = \text{insert } a (A \cap \{..<a\}) \rangle$  using bin ab aa by force

hence  $\langle \text{card } (A \cap \{..<?b\}) = \text{Suc } i \rangle$  using card by auto

thus  $\langle ?case \rangle$  using Suc.premis bin by auto

qed

**note**  $\langle \bigwedge \text{thesis}. ((\bigwedge a. a \in A \implies \text{card } (A \cap \{..<a\}) = i \implies \text{thesis}) \implies \text{thesis}) \rangle$

}

**note**  $\text{ex} = \text{this}$

{

fix  $i$

obtain  $a$  where  $a: \langle a \in A \wedge \text{card } (A \cap \{..<a\}) = i \rangle$  using ex by blast

have  $\text{ina}: \langle ?f i \in A \rangle$  and  $\text{card}: \langle \text{card } (A \cap \{..<?f i\}) = i \rangle$  using LeastI[of  $\langle \lambda a. a \in A \wedge \text{card } (A \cap \{..<a\}) = i \rangle \langle a \rangle$ , OF a] by auto

obtain  $b$  where  $b: \langle b \in A \wedge \text{card } (A \cap \{..<b\}) = \text{Suc } i \rangle$  using ex by blast

```

  have inab: ⟨?f (Suc i) ∈ A⟩ and cardb: ⟨card (A ∩ {..f (Suc i)}) = Suc i⟩ using LeastI[of ⟨λ a. a ∈ A
∧ card (A ∩ {..a}) = Suc i⟩ ⟨b⟩, OF b] by auto
  have ⟨?f i < ?f (Suc i)⟩ proof (rule ccontr)
    assume ⟨¬ ?f i < ?f (Suc i)⟩
    hence ⟨A ∩ {..f (Suc i)} ⊆ A ∩ {..f i}⟩ by auto
    moreover have ⟨finite (A ∩ {..f i})⟩ by auto
    ultimately have ⟨card(A ∩ {..f (Suc i)}) ≤ card (A ∩ {..f i})⟩ by (metis (erased, lifting) card-mono)
    thus ⟨False⟩ using card cardb by auto
  qed
  note this ina
}
note b = this
thus ⟨∀ i. ?f i < ?f (Suc i)⟩ by auto
have *: ⟨range ?f ⊆ A⟩ using b by auto
moreover
{
  fix a assume ina: ⟨a ∈ A⟩
  let ⟨?i⟩ = ⟨card (A ∩ {..a})⟩
  obtain b where b: ⟨b ∈ A ∧ card (A ∩ {..b}) = ?i⟩ using ex by blast
  have inab: ⟨?f ?i ∈ A⟩ and cardb: ⟨card (A ∩ {..f ?i}) = ?i⟩ using LeastI[of ⟨λ a. a ∈ A ∧ card (A
∩ {..a}) = ?i⟩ ⟨b⟩, OF b] by auto
  have le: ⟨?f ?i ≤ a⟩ using Least-le[of ⟨λ a. a ∈ A ∧ card (A ∩ {..a}) = ?i⟩ ⟨a⟩] ina by auto
  have ⟨a = ?f ?i⟩ proof (rule ccontr)
    have fin: ⟨finite (A ∩ {..a})⟩ by auto
    assume ⟨a ≠ ?f ?i⟩
    hence ⟨?f ?i < a⟩ using le by simp
    hence ⟨?f ?i ∈ A ∩ {..a}) using inab by auto
    moreover
    have ⟨A ∩ {..f ?i} ⊆ A ∩ {..a}) using le by auto
    hence ⟨A ∩ {..f ?i} = A ∩ {..a}) using cardb card-subset-eq[OF fin] by auto
    ultimately
    show ⟨False⟩ by auto
  qed
  hence ⟨a ∈ range ?f⟩ by auto
}
hence ⟨A ⊆ range ?f⟩ by auto
ultimately show ⟨range ?f = A⟩ by auto
qed

```

```

lemma mono-ge-id: ⟨∀ i. f i < f (Suc i) ⟹ i ≤ f i⟩
  apply (induction ⟨i⟩, auto)
  by (metis not-le not-less-eq-eq order-trans)

```

```

lemma insort-map-mono: assumes mono: ⟨∀ n m. n < m ⟶ f n < f m⟩ shows ⟨map f (insort n ns) =
insort (f n) (map f ns)⟩
  apply (induction ⟨ns⟩)
  apply auto
    apply (metis not-less not-less-iff-gr-or-eq mono)
    apply (metis antisym-conv1 less-imp-le mono)
  apply (metis mono not-less)
  by (metis mono not-less)

```

```

lemma sorted-list-of-set-map-mono: assumes mono: ⟨∀ n m. n < m ⟶ f n < f m⟩ and fin: ⟨finite A⟩
shows ⟨map f (sorted-list-of-set A) = sorted-list-of-set (f'A)⟩
using fin proof (induction)
  case empty thus ⟨?case⟩ by simp

```

**next**  
**case**  $(\text{insert } x \ A)$   
**have**  $[\text{simp}]: \langle \text{sorted-list-of-set } (\text{insert } x \ A) = \text{insort } x \ (\text{sorted-list-of-set } A) \rangle$  **using**  $\text{insert sorted-list-of-set-insert}$   
**by**  $\text{simp}$   
**have**  $\langle f \ ' \ \text{insert } x \ A = \text{insert } (f \ x) \ (f \ ' \ A) \rangle$  **by**  $\text{auto}$   
**moreover**  
**have**  $\langle f \ x \notin f \ ' \ A \rangle$  **apply**  $(\text{rule } \text{ccontr})$  **using**  $\text{insert}(2)$  **mono** **apply**  $\text{auto}$  **by**  $(\text{metis } \text{insert.hyps}(2) \ \text{mono } \text{neq-iff})$   
**ultimately**  
**have**  $\langle \text{sorted-list-of-set } (f \ ' \ \text{insert } x \ A) = \text{insort } (f \ x) \ (\text{sorted-list-of-set } (f \ ' \ A)) \rangle$  **using**  $\text{insert}(1)$   $\text{sorted-list-of-set-insert}$   
**by**  $\text{simp}$   
**also**  
**have**  $\langle \dots = \text{insort } (f \ x) \ (\text{map } f \ (\text{sorted-list-of-set } A)) \rangle$  **using**  $\text{insert.IH}$  **by**  $\text{auto}$   
**also have**  $\langle \dots = \text{map } f \ (\text{insort } x \ (\text{sorted-list-of-set } A)) \rangle$  **using**  $\text{insort-map-mono}[OF \ \text{mono}]$  **by**  $\text{auto}$   
**finally**  
**show**  $\langle \text{map } f \ (\text{sorted-list-of-set } (\text{insert } x \ A)) = \text{sorted-list-of-set } (f \ ' \ \text{insert } x \ A) \rangle$  **by**  $\text{simp}$   
**qed**

**lemma**  $\text{GreatestIB}$ :  
**fixes**  $n :: \langle \text{nat} \rangle$  **and**  $P$   
**assumes**  $a: \langle \exists k \leq n. P \ k \rangle$   
**shows**  $\text{GreatestBI}$ :  $\langle P \ (\text{GREATEST } k. k \leq n \wedge P \ k) \rangle$  **and**  $\text{GreatestB}$ :  $\langle (\text{GREATEST } k. k \leq n \wedge P \ k) \leq n \rangle$   
**proof** –  
**show**  $\langle P \ (\text{GREATEST } k. k \leq n \wedge P \ k) \rangle$  **using**  $\text{GreatestI-ex-nat}[OF \ \text{assms}]$  **by**  $\text{auto}$   
**show**  $\langle (\text{GREATEST } k. k \leq n \wedge P \ k) \leq n \rangle$  **using**  $\text{GreatestI-ex-nat}[OF \ \text{assms}]$  **by**  $\text{auto}$   
**qed**

**lemma**  $\text{GreatestB-le}$ :  
**fixes**  $n :: \langle \text{nat} \rangle$   
**assumes**  $\langle x \leq n \rangle$  **and**  $\langle P \ x \rangle$   
**shows**  $\langle x \leq (\text{GREATEST } k. k \leq n \wedge P \ k) \rangle$   
**proof** –  
**have**  $*$ :  $\langle \forall y. y \leq n \wedge P \ y \longrightarrow y < \text{Suc } n \rangle$  **by**  $\text{auto}$   
**then show**  $\langle x \leq (\text{GREATEST } k. k \leq n \wedge P \ k) \rangle$  **using**  $\text{assms}$  **by**  $(\text{blast } \text{intro: } \text{Greatest-le-nat})$   
**qed**

**lemma**  $\text{LeastBI-ex}$ : **assumes**  $\langle \exists k \leq n. P \ k \rangle$  **shows**  $\langle P \ (\text{LEAST } k::'c::\text{wellorder}. P \ k) \rangle$  **and**  $\langle (\text{LEAST } k. P \ k) \leq n \rangle$   
**proof** –  
**from**  $\text{assms}$  **obtain**  $k$  **where**  $k: k \leq n \ P \ k$  **by**  $\text{blast}$   
**thus**  $\langle P \ (\text{LEAST } k. P \ k) \rangle$  **using**  $\text{LeastI}[of \ \langle P \rangle \ \langle k \rangle]$  **by**  $\text{simp}$   
**show**  $\langle (\text{LEAST } k. P \ k) \leq n \rangle$  **using**  $\text{Least-le}[of \ \langle P \rangle \ \langle k \rangle]$   $k$  **by**  $\text{auto}$   
**qed**

**lemma**  $\text{allB-atLeastLessThan-lower}$ : **assumes**  $\langle (i::\text{nat}) \leq j \rangle$   $\langle \forall x \in \{i..<n\}. P \ x \rangle$  **shows**  $\langle \forall x \in \{j..<n\}. P \ x \rangle$   
**proof**  
**fix**  $x$  **assume**  $\langle x \in \{j..<n\} \rangle$  **hence**  $\langle x \in \{i..<n\} \rangle$  **using**  $\text{assms}(1)$  **by**  $\text{simp}$   
**thus**  $\langle P \ x \rangle$  **using**  $\text{assms}(2)$  **by**  $\text{auto}$   
**qed**

## 2.2 Facts about Paths

**context**  $\text{IFC}$   
**begin**

**lemma**  $\text{path0}$ :  $\langle \text{path } \sigma \ 0 = \text{entry} \rangle$  **unfolding**  $\text{path-def}$  **by**  $\text{auto}$

**lemma** *path-in-nodes*[intro]:  $\langle \text{path } \sigma \ k \in \text{nodes} \rangle$  **proof** (*induction*  $\langle k \rangle$ )  
**case** (*Suc*  $k$ )  
**hence**  $\langle \bigwedge \sigma'. (\text{path } \sigma \ k, \text{suc } (\text{path } \sigma \ k) \ \sigma') \in \text{edges} \rangle$  **by** *auto*  
**hence**  $\langle (\text{path } \sigma \ k, \text{path } \sigma \ (\text{Suc } k)) \in \text{edges} \rangle$  **unfolding** *path-def*  
**by** (*metis suc-def comp-apply funpow.simps(2) prod.collapse*)  
**thus**  $\langle ?\text{case} \rangle$  **using** *edges-nodes* **by** *force*  
**qed** (*auto simp add: path-def*)

**lemma** *path-is-path*[simp]:  $\langle \text{is-path } (\text{path } \sigma) \rangle$  **unfolding** *is-path-def path-def* **using** *step-suc-sem* **apply** *auto*  
**by** (*metis path-def suc-def edges-complete path-in-nodes prod.collapse*)

**lemma** *term-path-stable*: **assumes**  $\langle \text{is-path } \pi \rangle$   $\langle \pi \ i = \text{return} \rangle$  **and** *le*:  $\langle i \leq j \rangle$  **shows**  $\langle \pi \ j = \text{return} \rangle$   
**using** *le* **proof** (*induction*  $\langle j \rangle$ )  
**case** (*Suc*  $j$ )  
**show**  $\langle ?\text{case} \rangle$  **proof** *cases*  
**assume**  $\langle i \leq j \rangle$   
**hence**  $\langle \pi \ j = \text{return} \rangle$  **using** *Suc* **by** *simp*  
**hence**  $\langle (\text{return}, \pi \ (\text{Suc } j)) \in \text{edges} \rangle$  **using** *assms(1)* **unfolding** *is-path-def* **by** *metis*  
**thus**  $\langle \pi \ (\text{Suc } j) = \text{return} \rangle$  **using** *edges-return* **by** *auto*  
**next**  
**assume**  $\langle \neg i \leq j \rangle$   
**hence**  $\langle \text{Suc } j = i \rangle$  **using** *Suc* **by** *auto*  
**thus**  $\langle ?\text{thesis} \rangle$  **using** *assms(2)* **by** *auto*  
**qed**  
**next**  
**case** *0* **thus**  $\langle ?\text{case} \rangle$  **using** *assms* **by** *simp*  
**qed**

**lemma** *path-path-shift*: **assumes**  $\langle \text{is-path } \pi \rangle$  **shows**  $\langle \text{is-path } (\pi \ll m) \rangle$   
**using** *assms* **unfolding** *is-path-def* **by** *simp*

**lemma** *path-cons*: **assumes**  $\langle \text{is-path } \pi \rangle$   $\langle \text{is-path } \pi' \rangle$   $\langle \pi \ m = \pi' \ 0 \rangle$  **shows**  $\langle \text{is-path } (\pi \ @^m \ \pi') \rangle$   
**unfolding** *is-path-def* **proof**(*rule,cases*)  
**fix**  $n$  **assume**  $\langle m < n \rangle$  **thus**  $\langle ((\pi \ @^m \ \pi') \ n, (\pi \ @^m \ \pi') \ (\text{Suc } n)) \in \text{edges} \rangle$   
**using** *assms(2)* **unfolding** *is-path-def path-append-def*  
**by** (*auto,metis Suc-diff-Suc diff-Suc-Suc less-SucI*)  
**next**  
**fix**  $n$  **assume**  $\ast$ :  $\langle \neg m < n \rangle$  **thus**  $\langle ((\pi \ @^m \ \pi') \ n, (\pi \ @^m \ \pi') \ (\text{Suc } n)) \in \text{edges} \rangle$  **proof** *cases*  
**assume** [*simp*]:  $\langle n = m \rangle$   
**thus**  $\langle ?\text{thesis} \rangle$  **using** *assms* **unfolding** *is-path-def path-append-def* **by** *force*  
**next**  
**assume**  $\langle n \neq m \rangle$   
**hence**  $\langle \text{Suc } n \leq m \rangle$   $\langle n \leq m \rangle$  **using**  $\ast$  **by** *auto*  
**with** *assms(1)* **show**  $\langle ?\text{thesis} \rangle$  **unfolding** *is-path-def* **by** *auto*  
**qed**  
**qed**

**lemma** *is-path-loop*: **assumes**  $\langle \text{is-path } \pi \rangle$   $\langle 0 < i \rangle$   $\langle \pi \ i = \pi \ 0 \rangle$  **shows**  $\langle \text{is-path } (\lambda n. \pi \ (n \bmod i)) \rangle$  **unfolding**  
*is-path-def* **proof** (*rule,cases*)  
**fix**  $n$   
**assume**  $\langle 0 < \text{Suc } n \bmod i \rangle$   
**hence**  $\langle \text{Suc } n \bmod i = \text{Suc } (n \bmod i) \rangle$  **by** (*metis mod-Suc neq0-conv*)  
**moreover**  
**have**  $\langle (\pi \ (n \bmod i), \pi \ (\text{Suc } (n \bmod i))) \in \text{edges} \rangle$  **using** *assms(1)* **unfolding** *is-path-def* **by** *auto*  
**ultimately**

**show**  $\langle \pi (n \bmod i), \pi (Suc\ n \bmod i) \rangle \in edges$  **by simp**  
**next**  
**fix**  $n$   
**assume**  $\langle \neg 0 < Suc\ n \bmod i \rangle$   
**hence**  $\langle Suc\ n \bmod i = 0 \rangle$  **by auto**  
**moreover**  
**hence**  $\langle n \bmod i = i - 1 \rangle$  **using**  $assms(2)$  **by**  $(metis\ Zero\ neq\ Suc\ diff\ Suc\ 1\ mod\ Suc)$   
**ultimately**  
**show**  $\langle \pi(n \bmod i), \pi (Suc\ n \bmod i) \rangle \in edges$  **using**  $assms(1)$  **unfolding**  $is\ path\ def$  **by**  $(metis\ assms(3)\ mod\ Suc)$   
**qed**

**lemma**  $path\ nodes$ :  $\langle is\ path\ \pi \implies \pi\ k \in nodes \rangle$  **unfolding**  $is\ path\ def$  **using**  $edges\ nodes$  **by force**

**lemma**  $direct\ path\ return'$ : **assumes**  $\langle is\ path\ \pi \rangle \langle \pi\ 0 = x \rangle \langle x \neq return \rangle \langle \pi\ n = return \rangle$   
**obtains**  $\pi'\ n'$  **where**  $\langle is\ path\ \pi' \rangle \langle \pi'\ 0 = x \rangle \langle \pi'\ n' = return \rangle \langle \forall i > 0. \pi'\ i \neq x \rangle$   
**using**  $assms$  **proof**  $(induction\ \langle n \rangle\ arbitrary: \langle \pi \rangle\ rule: less\ induct)$   
**case**  $(less\ n\ \pi)$   
**hence**  $ih$ :  $\langle \bigwedge n' \pi'. n' < n \implies is\ path\ \pi' \implies \pi'\ 0 = x \implies \pi'\ n' = return \implies thesis \rangle$  **using**  $assms$  **by auto**  
**show**  $\langle thesis \rangle$  **proof cases**  
**assume**  $\langle \forall i > 0. \pi\ i \neq x \rangle$  **thus**  $\langle thesis \rangle$  **using**  $less$  **by auto**  
**next**  
**assume**  $\langle \neg (\forall i > 0. \pi\ i \neq x) \rangle$   
**then obtain**  $i$  **where**  $\langle 0 < i \rangle \langle \pi\ i = x \rangle$  **by auto**  
**hence**  $\langle (\pi \ll i)\ 0 = x \rangle$  **by auto**  
**moreover**  
**have**  $\langle i < n \rangle$  **using**  $less(3,5,6)$   $\langle \pi\ i = x \rangle$  **by**  $(metis\ linorder\ neqE\ nat\ term\ path\ stable\ less\ imp\ le)$   
**hence**  $\langle (\pi \ll i)\ (n - i) = return \rangle$  **using**  $less(6)$  **by auto**  
**moreover**  
**have**  $\langle is\ path\ (\pi \ll i) \rangle$  **using**  $less(3)$  **by**  $(metis\ path\ path\ shift)$   
**moreover**  
**have**  $\langle n - i < n \rangle$  **using**  $\langle 0 < i \rangle \langle i < n \rangle$  **by auto**  
**ultimately show**  $\langle thesis \rangle$  **using**  $ih$  **by auto**  
**qed**  
**qed**

**lemma**  $direct\ path\ return$ : **assumes**  $\langle x \in nodes \rangle \langle x \neq return \rangle$   
**obtains**  $\pi\ n$  **where**  $\langle is\ path\ \pi \rangle \langle \pi\ 0 = x \rangle \langle \pi\ n = return \rangle \langle \forall i > 0. \pi\ i \neq x \rangle$   
**using**  $direct\ path\ return'$   $[of\ \langle x \rangle]$   $reaching\ ret[OF\ assms(1)]$   $assms(2)$  **by blast**

**lemma**  $path\ append\ eq\ up\ to$ :  $\langle (\pi @^k \pi') =_k \pi \rangle$  **unfolding**  $eq\ up\ to\ def$  **by auto**

**lemma**  $eq\ up\ to\ le$ : **assumes**  $\langle k \leq n \rangle \langle \pi =_n \pi' \rangle$  **shows**  $\langle \pi =_k \pi' \rangle$  **using**  $assms$  **unfolding**  $eq\ up\ to\ def$  **by auto**

**lemma**  $eq\ up\ to\ refl$ : **shows**  $\langle \pi =_k \pi \rangle$  **unfolding**  $eq\ up\ to\ def$  **by auto**

**lemma**  $eq\ up\ to\ sym$ : **assumes**  $\langle \pi =_k \pi' \rangle$  **shows**  $\langle \pi' =_k \pi \rangle$  **using**  $assms$  **unfolding**  $eq\ up\ to\ def$  **by auto**

**lemma**  $eq\ up\ to\ apply$ : **assumes**  $\langle \pi =_k \pi' \rangle \langle j \leq k \rangle$  **shows**  $\langle \pi\ j = \pi'\ j \rangle$  **using**  $assms$  **unfolding**  $eq\ up\ to\ def$  **by auto**

**lemma**  $path\ swap\ ret$ : **assumes**  $\langle is\ path\ \pi \rangle$  **obtains**  $\pi'\ n$  **where**  $\langle is\ path\ \pi' \rangle \langle \pi =_k \pi' \rangle \langle \pi'\ n = return \rangle$   
**proof** –  
**have**  $nd$ :  $\langle \pi\ k \in nodes \rangle$  **using**  $assms\ path\ nodes$  **by simp**  
**obtain**  $\pi'\ n$  **where**  $*$ :  $\langle is\ path\ \pi' \rangle \langle \pi'\ 0 = \pi\ k \rangle \langle \pi'\ n = return \rangle$  **using**  $reaching\ ret[OF\ nd]$  **by blast**

**have**  $\langle \pi =_k (\pi @^k \pi') \rangle$  **by** (*metis eq-up-to-sym path-append-eq-up-to*)  
**moreover**  
**have**  $\langle \text{is-path } (\pi @^k \pi') \rangle$  **using** *assms \* path-cons* **by** *metis*  
**moreover**  
**have**  $\langle (\pi @^k \pi') (k + n) = \text{return} \rangle$  **using** *\** **by** *auto*  
**ultimately**  
**show**  $\langle \text{thesis} \rangle$  **using** *that* **by** *blast*  
**qed**

**lemma** *path-suc*:  $\langle \text{path } \sigma (Suc\ k) = \text{fst } (\text{step } (\text{path } \sigma\ k, \sigma^k)) \rangle$  **by** (*induction <k>, auto simp: path-def kth-state-def*)

**lemma** *kth-state-suc*:  $\langle \sigma^{Suc\ k} = \text{snd } (\text{step } (\text{path } \sigma\ k, \sigma^k)) \rangle$  **by** (*induction <k>, auto simp: path-def kth-state-def*)

## 2.3 Facts about Post Dominators

**lemma** *pd-trans*: **assumes** *1*:  $\langle y\ \text{pd} \rightarrow x \rangle$  **and** *2*:  $\langle z\ \text{pd} \rightarrow y \rangle$  **shows**  $\langle z\ \text{pd} \rightarrow x \rangle$

**proof** –

**{**  
**fix**  $\pi\ n$   
**assume**  $\exists[\text{simp}]: \langle \text{is-path } \pi \rangle \langle \pi\ 0 = x \rangle \langle \pi\ n = \text{return} \rangle$   
**then obtain**  $k$  **where**  $\langle \pi\ k = y \rangle$  **and**  $\langle k \leq n \rangle$  **using** *1* **unfolding** *is-pd-def* **by** *blast*  
**then have**  $\langle (\pi \ll k)\ 0 = y \rangle$  **and**  $\langle (\pi \ll k)\ (n - k) = \text{return} \rangle$  **by** *auto*  
**moreover have**  $\langle \text{is-path } (\pi \ll k) \rangle$  **by** (*metis 3(1) path-path-shift*)  
**ultimately obtain**  $k'$  **where**  $\langle (\pi \ll k)\ k' = z \rangle$  **and**  $\langle k' \leq n - k \rangle$  **using** *2* **unfolding** *is-pd-def* **by** *blast*  
**hence**  $\langle k + k' \leq n \rangle$  **and**  $\langle \pi\ (k + k') = z \rangle$  **using** *7* **by** *auto*  
**hence**  $\langle \exists k \leq n. \pi\ k = z \rangle$  **using** *path-nodes* **by** *auto*  
**}**  
**thus**  $\langle ?\text{thesis} \rangle$  **using** *1* **unfolding** *is-pd-def* **by** *blast*  
**qed**

**lemma** *pd-path*: **assumes**  $\langle y\ \text{pd} \rightarrow x \rangle$

**obtains**  $\pi\ n\ k$  **where**  $\langle \text{is-path } \pi \rangle$  **and**  $\langle \pi\ 0 = x \rangle$  **and**  $\langle \pi\ n = \text{return} \rangle$  **and**  $\langle \pi\ k = y \rangle$  **and**  $\langle k \leq n \rangle$   
**using** *assms* **unfolding** *is-pd-def* **using** *reaching-ret[of <x>]* **by** *blast*

**lemma** *pd-antisym*: **assumes** *xpdy*:  $\langle x\ \text{pd} \rightarrow y \rangle$  **and** *ypdx*:  $\langle y\ \text{pd} \rightarrow x \rangle$  **shows**  $\langle x = y \rangle$

**proof** –

**obtain**  $\pi\ n$  **where** *path*:  $\langle \text{is-path } \pi \rangle$  **and**  $\pi\ 0$ :  $\langle \pi\ 0 = x \rangle$  **and**  $\pi\ n$ :  $\langle \pi\ n = \text{return} \rangle$  **using** *pd-path[OF ypdx]* **by** *metis*

**hence** *kex*:  $\langle \exists k \leq n. \pi\ k = y \rangle$  **using** *ypdx* **unfolding** *is-pd-def* **by** *auto*

**obtain**  $k$  **where**  $k$ :  $\langle k = (\text{GREATEST } k. k \leq n \wedge \pi\ k = y) \rangle$  **by** *simp*

**have**  $\pi\ k$ :  $\langle \pi\ k = y \rangle$  **and**  $kn$ :  $\langle k \leq n \rangle$  **using**  $k$  *kex* **by** (*auto intro: GreatestIB*)

**have** *kpath*:  $\langle \text{is-path } (\pi \ll k) \rangle$  **by** (*metis path-path-shift path*)

**moreover have**  $k\ 0$ :  $\langle (\pi \ll k)\ 0 = y \rangle$  **using**  $\pi\ k$  **by** *simp*

**moreover have** *kreturn*:  $\langle (\pi \ll k)\ (n - k) = \text{return} \rangle$  **using**  $kn\ \pi\ n$  **by** *simp*

**ultimately have**  $ky'$ :  $\langle \exists k' \leq (n - k). (\pi \ll k)\ k' = x \rangle$  **using** *xpdy* **unfolding** *is-pd-def* **by** *simp*

**obtain**  $k'$  **where**  $k'$ :  $\langle k' = (\text{GREATEST } k'. k' \leq (n - k) \wedge (\pi \ll k)\ k' = x) \rangle$  **by** *simp*

**with**  $ky'$  **have**  $\pi\ k'$ :  $\langle (\pi \ll k)\ k' = x \rangle$  **and**  $kn'$ :  $\langle k' \leq (n - k) \rangle$  **by** (*auto intro: GreatestIB*)

**have**  $k'\text{path}$ :  $\langle \text{is-path } (\pi \ll k \ll k') \rangle$  **using** *kpath* **by** (*metis path-path-shift*)

**moreover have**  $k'\ 0$ :  $\langle (\pi \ll k \ll k')\ 0 = x \rangle$  **using**  $\pi\ k'$  **by** *simp*

**moreover have**  $k'\text{return}$ :  $\langle (\pi \ll k \ll k')\ (n - k - k') = \text{return} \rangle$  **using**  $kn'\ \text{kreturn}$  **by** (*metis path-shift-def le-add-diff-inverse*)

**ultimately have**  $ky''$ :  $\langle \exists k'' \leq (n - k - k'). (\pi \ll k \ll k')\ k'' = y \rangle$  **using** *ypdx* **unfolding** *is-pd-def* **by** *blast*

**obtain**  $k''$  **where**  $k''$ :  $\langle k'' = (\text{GREATEST } k''. k'' \leq (n - k - k') \wedge (\pi \ll k \ll k')\ k'' = y) \rangle$  **by** *simp*



with  $ky''$  have  $\pi k''$ :  $\langle \pi \langle k \langle k' \rangle \rangle k'' = y \rangle$  and  $kn''$ :  $\langle k'' \leq (n - k - k') \rangle$  by (auto intro: GreatestIB)

from  $this(1)$  have  $\langle \pi (k + k' + k'') = y \rangle$  by (metis path-shift-def add commute add.left-commute)

moreover

have  $\langle k + k' + k'' \leq n \rangle$  using  $kn'' kn' kn$  by simp

ultimately have  $\langle k + k' + k'' \leq k \rangle$  using  $k$  by (auto simp: GreatestB-le)

hence  $\langle k' = 0 \rangle$  by simp

with  $k0$   $\pi k'$  show  $\langle x = y \rangle$  by simp

qed

lemma  $pd-refl[simp]$ :  $\langle x \in nodes \implies x \text{ pd} \rightarrow x \rangle$  unfolding  $is-pd-def$  by blast

lemma  $pdt-trans-in-pdt$ :  $\langle (x,y) \in pdt^+ \implies (x,y) \in pdt \rangle$

proof (induction rule: trancl-induct)

case base thus  $\langle ?case \rangle$  by simp

next

case (step  $y z$ ) show  $\langle ?case \rangle$  unfolding  $pdt-def$  proof (simp)

have \*:  $\langle y \text{ pd} \rightarrow x \rangle \langle z \text{ pd} \rightarrow y \rangle$  using step unfolding  $pdt-def$  by auto

hence  $[simp]$ :  $\langle z \text{ pd} \rightarrow x \rangle$  using  $pd-trans$ [where  $x = \langle x \rangle$  and  $y = \langle y \rangle$  and  $z = \langle z \rangle$ ] by simp

have  $\langle x \neq z \rangle$  proof

assume  $\langle x = z \rangle$

hence  $\langle z \text{ pd} \rightarrow y \rangle \langle y \text{ pd} \rightarrow z \rangle$  using \* by auto

hence  $\langle z = y \rangle$  using  $pd-antisym$  by auto

thus  $\langle False \rangle$  using  $step(2)$  unfolding  $pdt-def$  by simp

qed

thus  $\langle x \neq z \wedge z \text{ pd} \rightarrow x \rangle$  by auto

qed

qed

lemma  $pdt-trancl-pdt$ :  $\langle pdt^+ = pdt \rangle$  using  $pdt-trans-in-pdt$  by fast

lemma  $trans-pdt$ :  $\langle trans \text{ pdt} \rangle$  by (metis  $pdt-trancl-pdt$   $trans-trancl$ )

definition  $[simp]$ :  $\langle pdt-inv = pdt^{-1} \rangle$

lemma  $wf-pdt-inv$ :  $\langle wf (pdt-inv) \rangle$  proof (rule ccontr)

assume  $\langle \neg wf (pdt-inv) \rangle$

then obtain  $f$  where  $\langle \forall i. (f (Suc i), f i) \in pdt^{-1} \rangle$  using  $wf-iff-no-infinite-down-chain$  by force

hence \*:  $\langle \forall i. (f i, f (Suc i)) \in pdt \rangle$  by simp

have \*\*:  $\langle \forall i. \forall j > i. (f i, f j) \in pdt \rangle$  proof (rule, rule, rule)

fix  $i j$  assume  $\langle i < (j::nat) \rangle$  thus  $\langle (f i, f j) \in pdt \rangle$  proof (induction  $\langle j \rangle$  rule: less-induct)

case (less  $k$ )

show  $\langle ?case \rangle$  proof (cases  $\langle Suc i < k \rangle$ )

case True

hence  $k: \langle k-1 < k \rangle \langle i < k-1 \rangle$  and  $sk: \langle Suc (k-1) = k \rangle$  by auto

show  $\langle ?thesis \rangle$  using  $less(1)$ [OF  $k$ ] \*[rule-format, of  $\langle k-1 \rangle$ , unfolded  $sk$ ]  $trans-pdt$ [unfolded  $trans-def$ ] by

blast

next

case False

hence  $\langle Suc i = k \rangle$  using  $less(2)$  by auto

then show  $\langle ?thesis \rangle$  using \* by auto

qed

qed

qed

hence \*\*\*:  $\langle \forall i. \forall j > i. f j \text{ pd} \rightarrow f i \rangle \langle \forall i. \forall j > i. f i \neq f j \rangle$  unfolding  $pdt-def$  by auto

hence \*\*\*\*:  $\langle \forall i > 0. f i \text{ pd} \rightarrow f 0 \rangle$  by simp

hence  $\langle f\ 0 \in \text{nodes} \rangle$  **using** \* *is-pd-def* **by** *fastforce*  
 then **obtain**  $\pi\ n$  **where**  $\pi: \langle \text{is-path } \pi \rangle \langle \pi\ 0 = f\ 0 \rangle \langle \pi\ n = \text{return} \rangle$  **using** *reaching-ret* **by** *blast*  
 hence  $\langle \forall\ i > 0. \exists\ k \leq n. \pi\ k = f\ i \rangle$  **using** *\*\*\*(1)*  $\langle f\ 0 \in \text{nodes} \rangle$  **unfolding** *is-pd-def* **by** *blast*  
 hence  $\pi f: \langle \forall\ i. \exists\ k \leq n. \pi\ k = f\ i \rangle$  **using**  $\pi(2)$  **by** (*metis le0 not-gr-zero*)  
 have  $\langle \text{range } f \subseteq \pi\ \{..n\} \rangle$  **proof**(*rule subsetI*)  
   **fix**  $x$  **assume**  $\langle x \in \text{range } f \rangle$   
   then **obtain**  $i$  **where**  $\langle x = f\ i \rangle$  **by** *auto*  
   then **obtain**  $k$  **where**  $\langle x = \pi\ k \rangle \langle k \leq n \rangle$  **using**  $\pi f$  **by** *metis*  
   **thus**  $\langle x \in \pi\ \{..n\} \rangle$  **by** *simp*  
**qed**  
 hence  $f: \langle \text{finite } (\text{range } f) \rangle$  **using** *finite-surj* **by** *auto*  
 hence  $f i: \langle \exists\ i. \text{infinite } \{j. f\ j = f\ i\} \rangle$  **using** *pigeonhole-infinite[OF - f]* **by** *auto*  
**obtain**  $i$  **where**  $\langle \text{infinite } \{j. f\ j = f\ i\} \rangle$  **using**  $f\ i$  **..**  
**thus**  $\langle \text{False} \rangle$   
**by** (*metis (mono-tags, lifting) \*\*\*(2) bounded-nat-set-is-finite gt-ex mem-Collect-eq nat-neq-iff*)  
**qed**

lemma *return-pd*: **assumes**  $\langle x \in \text{nodes} \rangle$  **shows**  $\langle \text{return } pd \rightarrow x \rangle$  **unfolding** *is-pd-def* **using** *assms* **by** *blast*

lemma *pd-total*: **assumes**  $xz: \langle x\ pd \rightarrow z \rangle$  **and**  $yz: \langle y\ pd \rightarrow z \rangle$  **shows**  $\langle x\ pd \rightarrow y \vee y\ pd \rightarrow x \rangle$

**proof** –

**obtain**  $\pi\ n$  **where**  $\text{path}: \langle \text{is-path } \pi \rangle$  **and**  $\pi 0: \langle \pi\ 0 = z \rangle$  **and**  $\pi n: \langle \pi\ n = \text{return} \rangle$  **using**  $xz$  *reaching-ret* **unfolding** *is-pd-def* **by** *force*

**have** \*:  $\langle \exists\ k \leq n. (\pi\ k = x \vee \pi\ k = y) \rangle$  (**is**  $\langle \exists\ k \leq n. ?P\ k \rangle$ ) **using**  $\text{path } \pi 0\ \pi n\ xz\ yz$  **unfolding** *is-pd-def* **by** *auto*

**obtain**  $k$  **where**  $k: \langle k = (\text{LEAST } k. \pi\ k = x \vee \pi\ k = y) \rangle$  **by** *simp*

**hence**  $kn: \langle k \leq n \rangle$  **and**  $\pi k: \langle \pi\ k = x \vee \pi\ k = y \rangle$  **using** *LeastBI-ex[OF \*]* **by** *auto*

**note**  $k\text{-le} = \text{Least-le}[\text{where } P = \langle ?P \rangle]$

**show**  $\langle ?thesis \rangle$  **proof** (*cases*)

**assume**  $kx: \langle \pi\ k = x \rangle$

**have**  $k\text{-min}: \langle \bigwedge\ k'. \pi\ k' = y \implies k \leq k' \rangle$  **using**  $k\text{-le}$  **unfolding**  $k$  **by** *auto*

{

**fix**  $\pi'$

**and**  $n' :: \langle \text{nat} \rangle$

**assume**  $\text{path}' : \langle \text{is-path } \pi' \rangle$  **and**  $\pi' 0: \langle \pi' 0 = x \rangle$  **and**  $\pi' n': \langle \pi' n' = \text{return} \rangle$

**have**  $\text{path}'' : \langle \text{is-path } (\pi\ @^k\ \pi') \rangle$  **using**  $\text{path-cons}[OF\ \text{path } \text{path}']\ kx\ \pi' 0$  **by** *auto*

**have**  $\pi'' 0: \langle (\pi\ @^k\ \pi')\ 0 = z \rangle$  **using**  $\pi 0$  **by** *simp*

**have**  $\pi'' n: \langle (\pi\ @^k\ \pi')\ (k+n') = \text{return} \rangle$  **using**  $\pi' n'\ kx\ \pi' 0$  **by** *auto*

**obtain**  $k'$  **where**  $k': \langle k' \leq k + n' \rangle \langle (\pi\ @^k\ \pi')\ k' = y \rangle$  **using**  $yz\ \text{path}''\ \pi'' 0\ \pi'' n$  **unfolding** *is-pd-def* **by**

*blast*

**have** \*\*:  $\langle k \leq k' \rangle$  **proof** (*rule ccontr*)

**assume**  $\langle \neg\ k \leq k' \rangle$

**hence**  $\langle k' < k \rangle$  **by** *simp*

**moreover**

**hence**  $\langle \pi\ k' = y \rangle$  **using**  $k'$  **by** *auto*

**ultimately**

**show**  $\langle \text{False} \rangle$  **using**  $k\text{-min}$  **by** *force*

**qed**

**hence**  $\langle \pi' (k' - k) = y \rangle$  **using**  $k'\ \pi' 0\ kx$  **by** *auto*

**moreover**

**have**  $\langle (k' - k) \leq n' \rangle$  **using**  $k'$  **by** *auto*

**ultimately**

**have**  $\langle \exists\ k \leq n'. \pi' k = y \rangle$  **by** *auto*

}

**hence**  $\langle y\ pd \rightarrow x \rangle$  **using**  $kx\ \text{path-nodes } \text{path}$  **unfolding** *is-pd-def* **by** *auto*

**thus**  $\langle ?thesis \rangle ..$   
**next** — This is analogous argument  
**assume**  $kx: \langle \pi k \neq x \rangle$   
**hence**  $ky: \langle \pi k = y \rangle$  **using**  $\pi k$  **by** *auto*  
**have**  $k\text{-min}: \langle \bigwedge k'. \pi k' = x \implies k \leq k' \rangle$  **using** *k-le unfolding*  $k$  **by** *auto*  
{  
  **fix**  $\pi'$   
  **and**  $n' :: \langle nat \rangle$   
  **assume**  $path': \langle is\text{-path } \pi' \rangle$  **and**  $\pi'0: \langle \pi' 0 = y \rangle$  **and**  $\pi'n': \langle \pi' n' = return \rangle$   
  **have**  $path'': \langle is\text{-path } (\pi @^k \pi') \rangle$  **using**  $path\text{-cons}[OF\ path\ path']$   $ky\ \pi'0$  **by** *auto*  
  **have**  $\pi''0: \langle (\pi @^k \pi') 0 = z \rangle$  **using**  $\pi 0$  **by** *simp*  
  **have**  $\pi''n: \langle (\pi @^k \pi') (k+n') = return \rangle$  **using**  $\pi'n'$   $ky\ \pi'0$  **by** *auto*  
  **obtain**  $k'$  **where**  $k': \langle k' \leq k + n' \rangle \langle (\pi @^k \pi') k' = x \rangle$  **using**  $xz\ path''\ \pi''0\ \pi''n$  **unfolding** *is-pd-def* **by**  
*blast*  
  **have**  $** : \langle k \leq k' \rangle$  **proof** (*rule ccontr*)  
  **assume**  $\langle \neg k \leq k' \rangle$   
  **hence**  $\langle k' < k \rangle$  **by** *simp*  
  **moreover**  
  **hence**  $\langle \pi k' = x \rangle$  **using**  $k'$  **by** *auto*  
  **ultimately**  
  **show**  $\langle False \rangle$  **using**  $k\text{-min}$  **by** *force*  
**qed**  
  **hence**  $\langle \pi' (k' - k) = x \rangle$  **using**  $k'\ \pi'0\ ky$  **by** *auto*  
  **moreover**  
  **have**  $\langle (k' - k) \leq n' \rangle$  **using**  $k'$  **by** *auto*  
  **ultimately**  
  **have**  $\langle \exists k \leq n'. \pi' k = x \rangle$  **by** *auto*  
}
**hence**  $\langle x\ pd \rightarrow y \rangle$  **using**  $ky\ path\text{-nodes}\ path$  **unfolding** *is-pd-def* **by** *auto*  
**thus**  $\langle ?thesis \rangle ..$   
**qed**  
**qed**

**lemma** *pds-finite*:  $\langle finite\ \{y . (x,y) \in pdt\} \rangle$  **proof** *cases*  
**assume**  $\langle x \in nodes \rangle$   
**then obtain**  $\pi\ n$  **where**  $\pi: \langle is\text{-path } \pi \rangle \langle \pi 0 = x \rangle \langle \pi n = return \rangle$  **using** *reaching-ret* **by** *blast*  
**have**  $*$ :  $\langle \forall y \in \{y. (x,y) \in pdt\}. y\ pd \rightarrow x \rangle$  **using** *pdt-def* **by** *auto*  
**have**  $\langle \forall y \in \{y. (x,y) \in pdt\}. \exists k \leq n. \pi k = y \rangle$  **using**  $*$   $\pi$  *is-pd-def* **by** *blast*  
**hence**  $\langle \{y. (x,y) \in pdt\} \subseteq \pi \text{ `` } \{..n\} \rangle$  **by** *auto*  
**then show**  $\langle ?thesis \rangle$  **using** *finite-surj* **by** *blast*  
**next**  
**assume**  $\langle \neg x \in nodes \rangle$   
**hence**  $\langle \{y. (x,y) \in pdt\} = \{\} \rangle$  **unfolding** *pdt-def is-pd-def* **using** *path-nodes reaching-ret* **by** *fastforce*  
**then show**  $\langle ?thesis \rangle$  **by** *simp*  
**qed**

**lemma** *ipd-exists*: **assumes** *node*:  $\langle x \in nodes \rangle$  **and** *not-ret*:  $\langle x \neq return \rangle$  **shows**  $\langle \exists y. y\ ipd \rightarrow x \rangle$   
**proof** —  
**let**  $\langle ?Q \rangle = \langle \{y. x \neq y \wedge y\ pd \rightarrow x\} \rangle$   
**have**  $*$ :  $\langle return \in ?Q \rangle$  **using** *assms return-pd* **by** *simp*  
**hence**  $**$ :  $\langle \exists x. x \in ?Q \rangle$  **by** *auto*  
**have** *fin*:  $\langle finite\ ?Q \rangle$  **using** *pds-finite unfolding pdt-def* **by** *auto*  
**have** *tot*:  $\langle \forall y\ z. y \in ?Q \wedge z \in ?Q \longrightarrow z\ pd \rightarrow y \vee y\ pd \rightarrow z \rangle$  **using** *pd-total* **by** *auto*  
**obtain**  $y$  **where** *ymax*:  $\langle y \in ?Q \rangle \langle \forall z \in ?Q. z = y \vee z\ pd \rightarrow y \rangle$  **using** *fin \*\* tot* **proof** (*induct*)  
  **case** *empty*

```

then show ⟨?case⟩ by auto
next
case (insert x F) show ⟨thesis⟩ proof (cases ⟨F = {}⟩)
  assume ⟨F = {}⟩
  thus ⟨thesis⟩ using insert(4)[of ⟨x⟩] by auto
next
assume ⟨F ≠ {}⟩
hence ⟨∃ x. x ∈ F⟩ by auto
have ⟨∧y. y ∈ F ⟹ ∃z ∈ F. z = y ∨ z pd→ y ⟹ thesis⟩ proof -
  fix y assume a: ⟨y ∈ F⟩ ⟨∃z ∈ F. z = y ∨ z pd→ y⟩
  have ⟨x ≠ y⟩ using insert a by auto
  have ⟨x pd→ y ∨ y pd→ x⟩ using insert(6) a(1) by auto
  thus ⟨thesis⟩ proof
    assume ⟨x pd→ y⟩
    hence ⟨∃z ∈ insert x F. z = y ∨ z pd→ y⟩ using a(2) by blast
    thus ⟨thesis⟩ using a(1) insert(4) by blast
  next
  assume ⟨y pd→ x⟩
  have ⟨∃z ∈ insert x F. z = x ∨ z pd→ x⟩ proof
    fix z assume ⟨z ∈ insert x F⟩ thus ⟨z = x ∨ z pd→ x⟩ proof(rule,simp)
      assume ⟨z ∈ F⟩
      hence ⟨z = y ∨ z pd→ y⟩ using a(2) by auto
      thus ⟨z = x ∨ z pd→ x⟩ proof(rule,simp add: ⟨y pd→ x⟩)
        assume ⟨z pd→ y⟩
        show ⟨z = x ∨ z pd→ x⟩ using ⟨y pd→ x⟩ ⟨z pd→ y⟩ pd-trans by blast
      qed
    qed
  qed
  then show ⟨thesis⟩ using insert by blast
  qed
  then show ⟨thesis⟩ using insert by blast
  qed
  then show ⟨thesis⟩ using insert by blast
  qed
hence ***: ⟨y pd→ x⟩ ⟨x ≠ y⟩ by auto
have ⟨∃ z. z ≠ x ∧ z pd→ x ⟶ z pd→ y⟩ proof (rule,rule)
  fix z
  assume a: ⟨z ≠ x ∧ z pd→ x⟩
  hence b: ⟨z ∈ ?Q⟩ by auto
  have ⟨y pd→ z ∨ z pd→ y⟩ using pd-total ***(1) a by auto
  thus ⟨z pd→ y⟩ proof
    assume c: ⟨y pd→ z⟩
    hence ⟨y = z⟩ using b ymax pdt-def pd-antisym by auto
    thus ⟨z pd→ y⟩ using c by simp
  qed simp
qed
with *** have ⟨y ipd→ x⟩ unfolding is-ipd-def by simp
thus ⟨?thesis⟩ by blast
qed

lemma ipd-unique: assumes yipd: ⟨y ipd→ x⟩ and y'ipd: ⟨y' ipd→ x⟩ shows ⟨y = y'⟩
proof -
  have 1: ⟨y pd→ y'⟩ and 2: ⟨y' pd→ y⟩ using yipd y'ipd unfolding is-ipd-def by auto
  show ⟨?thesis⟩ using pd-antisym[OF 1 2] .
qed

```

**lemma ipd-is-ipd:** **assumes**  $\langle x \in \text{nodes} \rangle$  **and**  $\langle x \neq \text{return} \rangle$  **shows**  $\langle \text{ipd } x \text{ ipd} \rightarrow x \rangle$  **proof** –  
**from** *assms* **obtain**  $y$  **where**  $\langle y \text{ ipd} \rightarrow x \rangle$  **using** *ipd-exists* **by** *auto*  
**moreover**  
**hence**  $\langle \bigwedge z. z \text{ ipd} \rightarrow x \implies z = y \rangle$  **using** *ipd-unique* **by** *simp*  
**ultimately show**  $\langle ?thesis \rangle$  **unfolding** *ipd-def* **by** (*auto intro: theI2*)  
**qed**

**lemma is-ipd-in-pdt:**  $\langle y \text{ ipd} \rightarrow x \implies (x, y) \in \text{pdt} \rangle$  **unfolding** *is-ipd-def pdt-def* **by** *auto*

**lemma ipd-in-pdt:**  $\langle x \in \text{nodes} \implies x \neq \text{return} \implies (x, \text{ipd } x) \in \text{pdt} \rangle$  **by** (*metis ipd-is-ipd is-ipd-in-pdt*)

**lemma no-pd-path:** **assumes**  $\langle x \in \text{nodes} \rangle$  **and**  $\langle \neg y \text{ pd} \rightarrow x \rangle$   
**obtains**  $\pi n$  **where**  $\langle \text{is-path } \pi \rangle$  **and**  $\langle \pi 0 = x \rangle$  **and**  $\langle \pi n = \text{return} \rangle$  **and**  $\langle \forall k \leq n. \pi k \neq y \rangle$   
**proof** (*rule ccontr*)  
**assume**  $\langle \neg \text{thesis} \rangle$   
**hence**  $\langle \forall \pi n. \text{is-path } \pi \wedge \pi 0 = x \wedge \pi n = \text{return} \longrightarrow (\exists k \leq n. \pi k = y) \rangle$  **using** *that* **by** *force*  
**thus**  $\langle \text{False} \rangle$  **using** *assms* **unfolding** *is-pd-def* **by** *auto*  
**qed**

**lemma pd-pd-ipd:** **assumes**  $\langle x \in \text{nodes} \rangle$   $\langle x \neq \text{return} \rangle$   $\langle y \neq x \rangle$   $\langle y \text{ pd} \rightarrow x \rangle$  **shows**  $\langle y \text{ pd} \rightarrow \text{ipd } x \rangle$   
**proof** –  
**have**  $\langle \text{ipd } x \text{ pd} \rightarrow x \rangle$  **by** (*metis assms(1,2) ipd-is-ipd is-ipd-def*)  
**hence**  $\langle y \text{ pd} \rightarrow \text{ipd } x \vee \text{ipd } x \text{ pd} \rightarrow y \rangle$  **by** (*metis assms(4) pd-total*)  
**thus**  $\langle ?thesis \rangle$  **proof**  
**have** 1:  $\langle \text{ipd } x \text{ ipd} \rightarrow x \rangle$  **by** (*metis assms(1,2) ipd-is-ipd*)  
**moreover**  
**assume**  $\langle \text{ipd } x \text{ pd} \rightarrow y \rangle$   
**ultimately**  
**show**  $\langle y \text{ pd} \rightarrow \text{ipd } x \rangle$  **unfolding** *is-ipd-def* **using** *assms(3,4)* **by** *auto*  
**qed** *auto*  
**qed**

**lemma pd-nodes:** **assumes**  $\langle y \text{ pd} \rightarrow x \rangle$  **shows** *pd-node1*:  $\langle y \in \text{nodes} \rangle$  **and** *pd-node2*:  $\langle x \in \text{nodes} \rangle$   
**proof** –  
**obtain**  $\pi k$  **where**  $\langle \text{is-path } \pi \rangle$   $\langle \pi k = y \rangle$  **using** *assms* **unfolding** *is-pd-def* **using** *reaching-ret* **by** *force*  
**thus**  $\langle y \in \text{nodes} \rangle$  **using** *path-nodes* **by** *auto*  
**show**  $\langle x \in \text{nodes} \rangle$  **using** *assms* **unfolding** *is-pd-def* **by** *simp*  
**qed**

**lemma pd-ret-is-ret:**  $\langle x \text{ pd} \rightarrow \text{return} \implies x = \text{return} \rangle$  **by** (*metis pd-antisym pd-node1 return-pd*)

**lemma ret-path-none-pd:** **assumes**  $\langle x \in \text{nodes} \rangle$   $\langle x \neq \text{return} \rangle$   
**obtains**  $\pi n$  **where**  $\langle \text{is-path } \pi \rangle$   $\langle \pi 0 = x \rangle$   $\langle \pi n = \text{return} \rangle$   $\langle \forall i > 0. \neg x \text{ pd} \rightarrow \pi i \rangle$   
**proof**(*rule ccontr*)  
**assume**  $\langle \neg \text{thesis} \rangle$   
**hence** \*:  $\langle \bigwedge \pi n. [\text{is-path } \pi; \pi 0 = x; \pi n = \text{return}] \implies \exists i > 0. x \text{ pd} \rightarrow \pi i \rangle$  **using** *that* **by** *blast*  
**obtain**  $\pi n$  **where** \*\*:  $\langle \text{is-path } \pi \rangle$   $\langle \pi 0 = x \rangle$   $\langle \pi n = \text{return} \rangle$   $\langle \forall i > 0. \pi i \neq x \rangle$  **using** *direct-path-return[OF assms]* **by** *metis*  
**then obtain**  $i$  **where** \*\*\*:  $\langle i > 0 \rangle$   $\langle x \text{ pd} \rightarrow \pi i \rangle$  **using** \* **by** *blast*  
**hence**  $\langle \pi i \neq \text{return} \rangle$  **using** *pd-ret-is-ret assms(2)* **by** *auto*  
**hence**  $\langle i < n \rangle$  **using** *assms(2) term-path-stable \*\** **by** (*metis linorder-neqE-nat less-imp-le*)  
**hence**  $\langle (\pi \ll i)(n-i) = \text{return} \rangle$  **using** \*\**(3)* **by** *auto*  
**moreover**  
**have**  $\langle (\pi \ll i) 0 = \pi i \rangle$  **by** *simp*  
**moreover**  
**have**  $\langle \text{is-path } (\pi \ll i) \rangle$  **using** \*\**(1)* *path-path-shift* **by** *metis*

ultimately  
obtain  $k$  where  $\langle \pi \ll i \rangle k = x \rangle$  using  $*** (2)$  unfolding *is-pd-def* by *metis*  
hence  $\langle \pi (i + k) = x \rangle$  by *auto*  
thus  $\langle False \rangle$  using  $** (4)$   $\langle i > 0 \rangle$  by *auto*  
qed

lemma *path-pd-ipd0'*: assumes  $\langle is-path \pi \rangle$  and  $\langle \pi n \neq return \rangle$   $\langle \pi n \neq \pi 0 \rangle$  and  $\langle \pi n pd \rightarrow \pi 0 \rangle$   
obtains  $k$  where  $\langle k \leq n \rangle$  and  $\langle \pi k = ipd(\pi 0) \rangle$

proof (rule *ccontr*)

have  $*$ :  $\langle \pi n pd \rightarrow ipd(\pi 0) \rangle$  by (metis *is-pd-def* *assms(3,4)* *pd-pd-ipd* *pd-ret-is-ret*)  
obtain  $\pi' n'$  where  $*$ :  $\langle is-path \pi' \rangle$   $\langle \pi' 0 = \pi n \rangle$   $\langle \pi' n' = return \rangle$   $\langle \forall i > 0. \neg \pi n pd \rightarrow \pi' i \rangle$  by (metis *assms(2)* *assms(4)* *pd-node1* *ret-path-none-pd*)  
hence  $\langle \forall i > 0. \pi' i \neq ipd(\pi 0) \rangle$  using  $*$  by *metis*  
moreover  
assume  $\langle \neg thesis \rangle$   
hence  $\langle \forall k \leq n. \pi k \neq ipd(\pi 0) \rangle$  using *that* by *blast*  
ultimately  
have  $\langle \forall i. (\pi @^n \pi') i \neq ipd(\pi 0) \rangle$  by (metis *diff-is-0-eq* *neq0-conv* *path-append-def*)  
moreover  
have  $\langle (\pi @^n \pi') (n + n') = return \rangle$   
by (metis  $\langle \pi' 0 = \pi n \rangle$   $\langle \pi' n' = return \rangle$  *add-diff-cancel-left'* *assms(2)* *diff-is-0-eq* *path-append-def*)  
moreover  
have  $\langle (\pi @^n \pi') 0 = \pi 0 \rangle$  by (metis *le0* *path-append-def*)  
moreover  
have  $\langle is-path (\pi @^n \pi') \rangle$  by (metis  $\langle \pi' 0 = \pi n \rangle$   $\langle is-path \pi' \rangle$  *assms(1)* *path-cons*)  
moreover  
have  $\langle ipd(\pi 0) pd \rightarrow \pi 0 \rangle$  by (metis  $** (2,3,4)$  *assms(2)* *assms(4)* *ipd-is-ipd* *is-ipd-def* *neq0-conv* *pd-node2*)  
moreover  
have  $\langle \pi 0 \in nodes \rangle$  by (metis *assms(1)* *path-nodes*)  
ultimately  
show  $\langle False \rangle$  unfolding *is-pd-def* by *blast*

qed

lemma *path-pd-ipd0*: assumes  $\langle is-path \pi \rangle$  and  $\langle \pi 0 \neq return \rangle$   $\langle \pi n \neq \pi 0 \rangle$  and  $\langle \pi n pd \rightarrow \pi 0 \rangle$   
obtains  $k$  where  $\langle k \leq n \rangle$  and  $\langle \pi k = ipd(\pi 0) \rangle$

proof cases

assume  $*$ :  $\langle \pi n = return \rangle$   
have  $\langle ipd(\pi 0) pd \rightarrow (\pi 0) \rangle$  by (metis *is-ipd-def* *is-pd-def* *assms(2,4)* *ipd-is-ipd*)  
with *assms(1,2,3)*  $*$  show  $\langle thesis \rangle$  unfolding *is-pd-def* by (metis *that*)

next

assume  $\langle \pi n \neq return \rangle$   
from *path-pd-ipd0'* [*OF* *assms(1)* *this* *assms(3,4)*] that show  $\langle thesis \rangle$  by *auto*

qed

lemma *path-pd-ipd*: assumes  $\langle is-path \pi \rangle$  and  $\langle \pi k \neq return \rangle$   $\langle \pi n \neq \pi k \rangle$  and  $\langle \pi n pd \rightarrow \pi k \rangle$  and *kn*:  $\langle k < n \rangle$

obtains  $l$  where  $\langle k < l \rangle$  and  $\langle l \leq n \rangle$  and  $\langle \pi l = ipd(\pi k) \rangle$

proof –

have  $\langle is-path (\pi \ll k) \rangle$   $\langle (\pi \ll k) 0 \neq return \rangle$   $\langle (\pi \ll k) (n - k) \neq (\pi \ll k) 0 \rangle$   $\langle (\pi \ll k) (n - k) pd \rightarrow (\pi \ll k) 0 \rangle$   
using *assms* *path-path-shift* by *auto*  
with *path-pd-ipd0* [*of*  $\langle \pi \ll k \rangle$   $\langle n - k \rangle$ ]  
obtain  $ka$  where  $\langle ka \leq n - k \rangle$   $\langle (\pi \ll k) ka = ipd((\pi \ll k) 0) \rangle$  .  
hence  $\langle k + ka \leq n \rangle$   $\langle \pi (k + ka) = ipd(\pi k) \rangle$  using *kn* by *auto*  
moreover  
hence  $\langle \pi (k + ka) ipd \rightarrow \pi k \rangle$  by (metis *assms(1)* *assms(2)* *ipd-is-ipd* *path-nodes*)  
hence  $\langle k < k + ka \rangle$  unfolding *is-ipd-def* by (metis *nat-neq-iff* *not-add-less1*)

ultimately  
 show  $\langle thesis \rangle$  using that[*of*  $\langle k+ka \rangle$ ] by auto  
 qed

lemma *path-ret-ipd*: assumes  $\langle is-path \pi \rangle$  and  $\langle \pi k \neq return \rangle$   $\langle \pi n = return \rangle$   
 obtains  $l$  where  $\langle k < l \rangle$  and  $\langle l \leq n \rangle$  and  $\langle \pi l = ipd(\pi k) \rangle$

proof –

have  $\langle \pi n \neq \pi k \rangle$  using *assms* by auto

moreover

have  $\langle k \leq n \rangle$  apply (rule *ccontr*) using *term-path-stable assms* by auto

hence  $\langle k < n \rangle$  by (*metis assms(2,3) dual-order.order-iff-strict*)

moreover

have  $\langle \pi n pd \rightarrow \pi k \rangle$  by (*metis assms(1,3) path-nodes return-pd*)

ultimately

obtain  $l$  where  $\langle k < l \rangle$   $\langle l \leq n \rangle$   $\langle \pi l = ipd(\pi k) \rangle$  using *assms path-pd-ipd* by *blast*

thus  $\langle thesis \rangle$  using that by auto

qed

lemma *pd-intro*: assumes  $\langle l pd \rightarrow k \rangle$   $\langle is-path \pi \rangle$   $\langle \pi 0 = k \rangle$   $\langle \pi n = return \rangle$   
 obtains  $i$  where  $\langle i \leq n \rangle$   $\langle \pi i = l \rangle$  using *assms unfolding is-pd-def* by *metis*

lemma *path-pd-pd0*: assumes *path*:  $\langle is-path \pi \rangle$  and *lpdn*:  $\langle \pi l pd \rightarrow n \rangle$  and *npd0*:  $\langle n pd \rightarrow \pi 0 \rangle$   
 obtains  $k$  where  $\langle k \leq l \rangle$   $\langle \pi k = n \rangle$

proof (rule *ccontr*)

assume  $\langle \neg thesis \rangle$

hence *notn*:  $\langle \bigwedge k. k \leq l \implies \pi k \neq n \rangle$  using that by *blast*

have *nret*:  $\langle \pi l \neq return \rangle$  by (*metis is-pd-def assms(1,3) notn*)

obtain  $\pi' n'$  where *path'*:  $\langle is-path \pi' \rangle$  and  $\pi 0'$ :  $\langle \pi' 0 = \pi l \rangle$  and  $\pi n'$ :  $\langle \pi' n' = return \rangle$  and *nonepd*:  $\langle \forall i > 0. \neg \pi l pd \rightarrow \pi' i \rangle$

using *nret path path-nodes ret-path-none-pd* by *metis*

have  $\langle \pi l \neq n \rangle$  using *notn* by *simp*

hence  $\langle \forall i. \pi' i \neq n \rangle$  using *nonepd*  $\pi 0'$  *lpdn* by (*metis neq0-conv*)

hence *notn'*:  $\langle \forall i. (\pi @^l \pi') i \neq n \rangle$  using *notn*  $\pi 0'$  by auto

have  $\langle is-path (\pi @^l \pi') \rangle$  using *path path'* by (*metis*  $\pi 0'$  *path-cons*)

moreover

have  $\langle (\pi @^l \pi') 0 = \pi 0 \rangle$  by *simp*

moreover

have  $\langle (\pi @^l \pi') (n' + l) = return \rangle$  using  $\pi 0'$   $\pi n'$  by auto

ultimately

show  $\langle False \rangle$  using *notn' npd0 unfolding is-pd-def* by *blast*

qed

## 2.4 Facts about Control Dependencies

lemma *icd-imp-cd*:  $\langle n icd^\pi \rightarrow k \implies n cd^\pi \rightarrow k \rangle$  by (*metis is-icdi-def*)

lemma *ipd-impl-not-cd*: assumes  $\langle j \in \{k..i\} \rangle$  and  $\langle \pi j = ipd(\pi k) \rangle$  shows  $\langle \neg i cd^\pi \rightarrow k \rangle$   
 by (*metis assms(1) assms(2) is-cdi-def*)

lemma *cd-not-ret*: assumes  $\langle i cd^\pi \rightarrow k \rangle$  shows  $\langle \pi k \neq return \rangle$  by (*metis is-cdi-def assms nat-less-le term-path-stable*)

lemma *cd-path-shift*: assumes  $\langle j \leq k \rangle$   $\langle is-path \pi \rangle$  shows  $\langle (i cd^\pi \rightarrow k) = (i - j cd^\pi \langle^j \rightarrow k - j \rangle) \rangle$  proof

**assume**  $a: \langle i \text{ cd}^\pi \rightarrow k \rangle$   
**hence**  $b: \langle k < i \rangle$  **by** (*metis is-cdi-def*)  
**hence**  $\langle \text{is-path } (\pi \ll j) \rangle \langle k - j < i - j \rangle$  **using** *assms apply* (*metis path-path-shift*)  
**by** (*metis assms(1) b diff-less-mono*)  
**moreover**  
**have**  $c: \langle \forall j \in \{k..i\}, \pi j \neq \text{ipd } (\pi k) \rangle$  **by** (*metis a ipd-impl-not-cd*)  
**hence**  $\langle \forall ja \in \{k - j..i - j\}, (\pi \ll j) ja \neq \text{ipd } ((\pi \ll j) (k - j)) \rangle$  **using** *b assms by auto fastforce*  
**moreover**  
**have**  $\langle j < i \rangle$  **using** *assms(1) b by auto*  
**hence**  $\langle (\pi \ll j) (i - j) \neq \text{return} \rangle$  **using** *a unfolding is-cdi-def by auto*  
**ultimately**  
**show**  $\langle i - j \text{ cd}^{\pi \ll j} \rightarrow k - j \rangle$  **unfolding** *is-cdi-def by simp*  
**next**  
**assume**  $a: \langle i - j \text{ cd}^{\pi \ll j} \rightarrow k - j \rangle$   
**hence**  $b: \langle k - j < i - j \rangle$  **by** (*metis is-cdi-def*)  
**moreover**  
**have**  $c: \langle \forall ja \in \{k - j..i - j\}, (\pi \ll j) ja \neq \text{ipd } ((\pi \ll j) (k - j)) \rangle$  **by** (*metis a ipd-impl-not-cd*)  
**have**  $\langle \forall j \in \{k..i\}, \pi j \neq \text{ipd } (\pi k) \rangle$  **proof** (*rule,goal-cases*) **case** (*1 n*)  
**hence**  $\langle n - j \in \{k - j..i - j\} \rangle$  **using** *assms by auto*  
**hence**  $\langle \pi (j + (n - j)) \neq \text{ipd } (\pi (j + (k - j))) \rangle$  **by** (*metis c path-shift-def*)  
**thus**  $\langle ?\text{case} \rangle$  **using** *1 assms(1) by auto*  
**qed**  
**moreover**  
**have**  $\langle j < i \rangle$  **using** *assms(1) b by auto*  
**hence**  $\langle \pi i \neq \text{return} \rangle$  **using** *a unfolding is-cdi-def by auto*  
**ultimately**  
**show**  $\langle i \text{ cd}^\pi \rightarrow k \rangle$  **unfolding** *is-cdi-def by (metis assms(1) assms(2) diff-is-0-eq' le-diff-iff nat-le-linear nat-less-le)*  
**qed**

**lemma** *cd-path-shift0*: **assumes**  $\langle \text{is-path } \pi \rangle$  **shows**  $\langle (i \text{ cd}^\pi \rightarrow k) = (i - k \text{ cd}^{\pi \ll k} \rightarrow 0) \rangle$   
**using** *cd-path-shift[OF - assms]* **by** (*metis diff-self-eq-0 le-refl*)

**lemma** *icd-path-shift*: **assumes**  $\langle l \leq k \rangle \langle \text{is-path } \pi \rangle$  **shows**  $\langle (i \text{ icd}^\pi \rightarrow k) = (i - l \text{ icd}^{\pi \ll l} \rightarrow k - l) \rangle$   
**proof** –  
**have**  $\langle \text{is-path } (\pi \ll l) \rangle$  **using** *path-path-shift assms(2) by auto*  
**moreover**  
**have**  $\langle (i \text{ cd}^\pi \rightarrow k) = (i - l \text{ cd}^{\pi \ll l} \rightarrow k - l) \rangle$  **using** *assms cd-path-shift by auto*  
**moreover**  
**have**  $\langle (\forall m \in \{k <..< i\}, \neg i \text{ cd}^\pi \rightarrow m) = (\forall m \in \{k - l <..< i - l\}, \neg i - l \text{ cd}^{\pi \ll l} \rightarrow m) \rangle$   
**proof** –  
**{** **fix**  $m$  **assume**  $*$ :  $\langle \forall m \in \{k - l <..< i - l\}, \neg i - l \text{ cd}^{\pi \ll l} \rightarrow m \rangle \langle m \in \{k <..< i\} \rangle$   
**hence**  $\langle m - l \in \{k - l <..< i - l\} \rangle$  **using** *assms(1) by auto*  
**hence**  $\langle \neg i - l \text{ cd}^{\pi \ll l} \rightarrow (m - l) \rangle$  **using**  $*$  **by** *blast*  
**moreover**  
**have**  $\langle l \leq m \rangle$  **using**  $*$  *assms by auto*  
**ultimately** **have**  $\langle \neg i \text{ cd}^\pi \rightarrow m \rangle$  **using** *assms(2) cd-path-shift by blast*  
**}**  
**moreover**  
**{** **fix**  $m$  **assume**  $*$ :  $\langle \forall m \in \{k <..< i\}, \neg i \text{ cd}^\pi \rightarrow m \rangle \langle m - l \in \{k - l <..< i - l\} \rangle$   
**hence**  $\langle m \in \{k <..< i\} \rangle$  **using** *assms(1) by auto*  
**hence**  $\langle \neg i \text{ cd}^\pi \rightarrow m \rangle$  **using**  $*$  **by** *blast*  
**moreover**  
**have**  $\langle l \leq m \rangle$  **using**  $*$  *assms by auto*  
**ultimately** **have**  $\langle \neg i - l \text{ cd}^{\pi \ll l} \rightarrow (m - l) \rangle$  **using** *assms(2) cd-path-shift by blast*  
**}**



ultimately show  $\langle ?thesis \rangle$  by auto (metis diff-add-inverse)  
qed  
ultimately  
show  $\langle ?thesis \rangle$  unfolding is-icdi-def using assms by blast  
qed

lemma icd-path-shift0: assumes  $\langle is-path \pi \rangle$  shows  $\langle (i \text{ icd}^\pi \rightarrow k) = (i - k \text{ icd}^{\pi \ll k} \rightarrow 0) \rangle$   
using icd-path-shift[OF - assms] by (metis diff-self-eq-0 le-refl)

lemma cdi-path-swap: assumes  $\langle is-path \pi' \rangle \langle j \text{ cd}^\pi \rightarrow k \rangle \langle \pi =_j \pi' \rangle$  shows  $\langle j \text{ cd}^{\pi'} \rightarrow k \rangle$  using assms unfolding eq-up-to-def is-cdi-def by auto

lemma cdi-path-swap-le: assumes  $\langle is-path \pi' \rangle \langle j \text{ cd}^\pi \rightarrow k \rangle \langle \pi =_n \pi' \rangle \langle j \leq n \rangle$  shows  $\langle j \text{ cd}^{\pi'} \rightarrow k \rangle$  by (metis assms cdi-path-swap eq-up-to-le)

lemma not-cd-impl-ipd: assumes  $\langle is-path \pi \rangle$  and  $\langle k < i \rangle$  and  $\langle \neg i \text{ cd}^\pi \rightarrow k \rangle$  and  $\langle \pi i \neq \text{return} \rangle$  obtains  $j$  where  $\langle j \in \{k..i\} \rangle$  and  $\langle \pi j = \text{ipd}(\pi k) \rangle$   
by (metis assms(1) assms(2) assms(3) assms(4) is-cdi-def)

lemma icd-is-the-icd: assumes  $\langle i \text{ icd}^\pi \rightarrow k \rangle$  shows  $\langle k = (\text{THE } k. i \text{ icd}^\pi \rightarrow k) \rangle$  using assms icd-uniq  
by (metis the1-equality)

lemma all-ipd-imp-ret: assumes  $\langle is-path \pi \rangle$  and  $\langle \forall i. \pi i \neq \text{return} \longrightarrow (\exists j > i. \pi j = \text{ipd}(\pi i)) \rangle$  shows  $\langle \exists j. \pi j = \text{return} \rangle$

proof –

{ fix  $x$  assume \*:  $\langle \pi 0 = x \rangle$

have  $\langle ?thesis \rangle$  using wf-pdt-inv \* assms

proof(induction  $\langle x \rangle$  arbitrary:  $\langle \pi \rangle$  rule: wf-induct-rule )

case (less  $x \pi$ ) show  $\langle ?case \rangle$  proof (cases  $\langle x = \text{return} \rangle$ )

case True thus  $\langle ?thesis \rangle$  using less(2) by auto

next

assume not-ret:  $\langle x \neq \text{return} \rangle$

moreover

then obtain  $k$  where  $k\text{-ipd}$ :  $\langle \pi k = \text{ipd } x \rangle$  using less(2,4) by auto

moreover

have  $\langle x \in \text{nodes} \rangle$  using less(2,3) by (metis path-nodes)

ultimately

have  $\langle (x, \pi k) \in \text{pdt} \rangle$  by (metis ipd-in-pdt)

hence  $a$ :  $\langle (\pi k, x) \in \text{pdt-inv} \rangle$  unfolding pdt-inv-def by simp

have  $b$ :  $\langle is-path(\pi \ll k) \rangle$  by (metis less.prem(2) path-path-shift)

have  $c$ :  $\langle \forall i. (\pi \ll k) i \neq \text{return} \longrightarrow (\exists j > i. (\pi \ll k) j = \text{ipd}((\pi \ll k) i)) \rangle$  using less(4) apply auto

by (metis (full-types) ab-semigroup-add-class.add-ac(1) less-add-same-cancel1 less-imp-add-positive)

from less(1)[OF  $a - b c$ ]

have  $\langle \exists j. (\pi \ll k) j = \text{return} \rangle$  by auto

thus  $\langle \exists j. \pi j = \text{return} \rangle$  by auto

qed

qed

}

thus  $\langle ?thesis \rangle$  by simp

qed

lemma loop-has-cd: assumes  $\langle is-path \pi \rangle \langle 0 < i \rangle \langle \pi i = \pi 0 \rangle \langle \pi 0 \neq \text{return} \rangle$  shows  $\langle \exists k < i. i \text{ cd}^\pi \rightarrow k \rangle$   
proof (rule ccontr)

let  $\langle ?\pi \rangle = \langle \lambda n. \pi (n \text{ mod } i) \rangle$

assume  $\langle \neg (\exists k < i. i \text{ cd}^\pi \rightarrow k) \rangle$

hence  $\langle \forall k < i. \neg i \text{ cd}^\pi \rightarrow k \rangle$  by blast

**hence**  $\ast$ :  $\langle \forall k < i. (\exists j \in \{k..i\}. \pi j = \text{ipd } (\pi k)) \rangle$  **using**  $\text{assms}(1,3,4)$  **not-cd-impl-ipd** **by**  $\text{metis}$   
**have**  $\langle \forall k. (\exists j > k. ?\pi j = \text{ipd } (? \pi k)) \rangle$  **proof**  
**fix**  $k$   
**have**  $\langle k \bmod i < i \rangle$  **using**  $\text{assms}(2)$  **by**  $\text{auto}$   
**with**  $\ast$  **obtain**  $j$  **where**  $\langle j \in \{(k \bmod i)..i\} \langle \pi j = \text{ipd } (\pi (k \bmod i)) \rangle$  **by**  $\text{auto}$   
**then obtain**  $j'$  **where**  $1$ :  $\langle j' < i \rangle \langle \pi j' = \text{ipd } (\pi (k \bmod i)) \rangle$   
**by**  $(\text{cases } \langle j = i \rangle, \text{auto}, \text{metis } \text{assms}(2) \text{ assms}(3), \text{metis } \text{le-neq-implies-less})$   
**then obtain**  $j''$  **where**  $2$ :  $\langle j'' > k \rangle \langle j'' \bmod i = j' \rangle$  **by**  $(\text{metis } \text{mod-bound-instance})$   
**hence**  $\langle ?\pi j'' = \text{ipd } (? \pi k) \rangle$  **using**  $1$  **by**  $\text{auto}$   
**with**  $2(1)$   
**show**  $\langle \exists j > k. ?\pi j = \text{ipd } (? \pi k) \rangle$  **by**  $\text{auto}$   
**qed**  
**moreover**  
**have**  $\langle \text{is-path } ?\pi \rangle$  **by**  $(\text{metis } \text{assms}(1) \text{ assms}(2) \text{ assms}(3) \text{ is-path-loop})$   
**ultimately**  
**obtain**  $k$  **where**  $\langle ?\pi k = \text{return} \rangle$  **by**  $(\text{metis } (\text{lifting}) \text{ all-ipd-imp-ret})$   
**moreover**  
**have**  $\langle k \bmod i < i \rangle$  **by**  $(\text{simp add: } \text{assms}(2))$   
**ultimately**  
**have**  $\langle \pi i = \text{return} \rangle$  **by**  $(\text{metis } \text{assms}(1) \text{ term-path-stable less-imp-le})$   
**thus**  $\langle \text{False} \rangle$  **by**  $(\text{metis } \text{assms}(3) \text{ assms}(4))$   
**qed**

**lemma**  $\text{loop-has-cd}'$ : **assumes**  $\langle \text{is-path } \pi \rangle \langle j < i \rangle \langle \pi i = \pi j \rangle \langle \pi j \neq \text{return} \rangle$  **shows**  $\langle \exists k \in \{j..<i\}. i \text{ cd}^\pi \rightarrow k \rangle$

**proof** –

**have**  $\langle \exists k' < i - j. i - j \text{ cd}^\pi \langle j \rightarrow k' \rangle$   
**apply**  $(\text{rule } \text{loop-has-cd})$   
**apply**  $(\text{metis } \text{assms}(1) \text{ path-path-shift})$   
**apply**  $(\text{auto simp add: } \text{assms less-imp-le})$   
**done**  
**then obtain**  $k$  **where**  $k$ :  $\langle k < i - j \rangle \langle i - j \text{ cd}^\pi \langle j \rightarrow k \rangle$  **by**  $\text{auto}$   
**hence**  $k'$ :  $\langle (k + j) < i \rangle \langle i - j \text{ cd}^\pi \langle j \rightarrow (k + j) - j \rangle$  **by**  $\text{auto}$   
**note**  $\text{cd-path-shift}[OF - \text{assms}(1)]$   
**hence**  $\langle i \text{ cd}^\pi \rightarrow k + j \rangle$  **using**  $k'(2)$  **by**  $(\text{metis } \text{le-add1 add commute})$   
**with**  $k'(1)$  **show**  $\langle ?\text{thesis} \rangle$  **by**  $\text{force}$

**qed**

**lemma**  $\text{claim}''$ : **assumes**  $\text{path}\pi$ :  $\langle \text{is-path } \pi \rangle$  **and**  $\text{path}\pi'$ :  $\langle \text{is-path } \pi' \rangle$   
**and**  $\pi i$ :  $\langle \pi i = \pi' i' \rangle$  **and**  $\pi j$ :  $\langle \pi j = \pi' j' \rangle$

**and**  $\text{not-cd}$ :  $\langle \forall k. \neg j \text{ cd}^\pi \rightarrow k \rangle \langle \forall k. \neg i' \text{ cd}^{\pi'} \rightarrow k \rangle$

**and**  $\text{nret}$ :  $\langle \pi i \neq \text{return} \rangle$

**and**  $\text{ilj}$ :  $\langle i < j \rangle$

**shows**  $\langle i' < j' \rangle$  **proof**  $(\text{rule } \text{ccontr})$

**assume**  $\langle \neg i' < j' \rangle$

**hence**  $\text{jlei}$ :  $\langle j' \leq i' \rangle$  **by**  $\text{auto}$

**show**  $\langle \text{False} \rangle$  **proof**  $(\text{cases})$

**assume**  $\text{j'li'}$ :  $\langle j' < i' \rangle$

**define**  $\pi''$  **where**  $\langle \pi'' \equiv (\pi @^j (\pi' \langle j' \rangle)) \langle i \rangle$

**note**  $\pi''\text{-def}[\text{simp}]$

**have**  $\langle \pi j = (\pi' \langle j' \rangle) 0 \rangle$  **by**  $(\text{metis } \text{path-shift-def } \text{Nat.add-0-right } \pi j)$

**hence**  $\langle \text{is-path } \pi'' \rangle$  **using**  $\text{path}\pi \text{ path}\pi' \pi''\text{-def path-path-shift path-cons}$  **by**  $\text{presburger}$

**moreover**

**have**  $\langle \pi'' (j - i + (i' - j')) = \pi'' 0 \rangle$  **using**  $\text{ilj jlei } \pi i \pi j$

**by**  $(\text{auto}, \text{metis } \text{add-diff-cancel-left' le-antisym le-diff-conv le-eq-less-or-eq})$

**moreover**

**have**  $\langle \pi'' 0 \neq \text{return} \rangle$  **by** (*simp add: ilj less-or-eq-imp-le nret*)  
**moreover**  
**have**  $\langle 0 < j - i + (i' - j') \rangle$  **by** (*metis add-is-0 ilj neq0-conv zero-less-diff*)  
**ultimately obtain**  $k$  **where**  $k: \langle k < j - i + (i' - j') \rangle \langle j - i + (i' - j') \text{ cd}^{\pi''} \rightarrow k \rangle$  **by** (*metis loop-has-cd*)  
**hence**  $*$ :  $\langle \forall l \in \{k..j - i + (i' - j')\}. \pi'' l \neq \text{ipd} (\pi'' k) \rangle$  **by** (*metis is-cdi-def*)  
**show**  $\langle \text{False} \rangle$  **proof** (*cases*  $\langle k < j - i \rangle$ )  
  **assume**  $a: \langle k < j - i \rangle$   
  **hence**  $b: \langle \pi'' k = \pi (i + k) \rangle$  **by** *auto*  
  **have**  $\langle \forall l \in \{i + k..j\}. \pi l \neq \text{ipd} (\pi (i + k)) \rangle$  **proof**  
    **fix**  $l$  **assume**  $l: \langle l \in \{i + k..j\} \rangle$   
    **hence**  $\langle \pi l = \pi'' (l - i) \rangle$  **by** *auto*  
    **moreover**  
    **from**  $a$  **have**  $\langle l - i \in \{k .. j - i + (i' - j')\} \rangle$  **by** *force*  
    **ultimately show**  $\langle \pi l \neq \text{ipd} (\pi (i + k)) \rangle$  **using**  $*$   $b$  **by** *auto*  
  **qed**  
  **moreover**  
  **have**  $\langle i + k < j \rangle$  **using**  $a$  **by** *simp*  
  **moreover**  
  **have**  $\langle \pi j \neq \text{return} \rangle$  **by** (*metis*  $\pi i$   $\pi j$   $j'li'$  *nret path* $\pi'$  *term-path-stable less-imp-le*)  
  **ultimately**  
  **have**  $\langle j \text{ cd}^{\pi} \rightarrow i + k \rangle$  **by** (*metis not-cd-impl-ipd path* $\pi$ )  
  **thus**  $\langle \text{False} \rangle$  **by** (*metis not-cd(1)*)  
**next**  
  **assume**  $\langle \neg k < j - i \rangle$   
  **hence**  $a: \langle j - i \leq k \rangle$  **by** *simp*  
  **hence**  $b: \langle \pi'' k = \pi' (j' + (i + k) - j) \rangle$  **unfolding**  $\pi''$ -*def path-shift-def path-append-def* **using** *ilj*  
  **by** (*auto,metis*  $\pi j$  *add-diff-cancel-left' le-antisym le-diff-conv add commute*)  
  **have**  $\langle \forall l \in \{j' + (i + k) - j..i'\}. \pi' l \neq \text{ipd} (\pi' (j' + (i + k) - j)) \rangle$  **proof**  
    **fix**  $l$  **assume**  $l: \langle l \in \{j' + (i + k) - j..i'\} \rangle$   
    **hence**  $\langle \pi' l = \pi'' (j + l - i - j') \rangle$  **unfolding**  $\pi''$ -*def path-shift-def path-append-def* **using** *ilj*  
    **by** (*auto,metis* *Nat.diff-add-assoc*  $\pi j$  *a add commute add-diff-cancel-left' add-leD1 le-antisym le-diff-conv*)  
    **moreover**  
    **from**  $a$   $l$  **have**  $\langle j + l - i - j' \in \{k .. j - i + (i' - j')\} \rangle$  **by** *force*  
    **ultimately show**  $\langle \pi' l \neq \text{ipd} (\pi' (j' + (i + k) - j)) \rangle$  **using**  $*$   $b$  **by** *auto*  
  **qed**  
  **moreover**  
  **have**  $\langle j' + (i + k) - j < i' \rangle$  **using**  $a$   $j'li'$  *ilj k(1)* **by** *linarith*  
  **moreover**  
  **have**  $\langle \pi' i' \neq \text{return} \rangle$  **by** (*metis*  $\pi i$  *nret*)  
  **ultimately**  
  **have**  $\langle i' \text{ cd}^{\pi'} \rightarrow j' + (i + k) - j \rangle$  **by** (*metis not-cd-impl-ipd path* $\pi'$ )  
  **thus**  $\langle \text{False} \rangle$  **by** (*metis not-cd(2)*)  
**qed**  
**next**  
  **assume**  $\langle \neg j' < i' \rangle$   
  **hence**  $\langle j' = i' \rangle$  **by** (*metis*  $\langle \neg i' < j' \rangle$  *linorder-cases*)  
  **hence**  $\langle \pi i = \pi j \rangle$  **by** (*metis*  $\pi i$   $\pi j$ )  
  **thus**  $\langle \text{False} \rangle$  **by** (*metis* *ilj loop-has-cd' not-cd(1) nret path* $\pi$ )  
**qed**  
**qed**  
  
**lemma** *other-claim'*: **assumes** *path*:  $\langle \text{is-path } \pi \rangle$  **and** *eq*:  $\langle \pi i = \pi j \rangle$  **and**  $\langle \pi i \neq \text{return} \rangle$   
**and** *icd*:  $\langle \forall k. \neg i \text{ cd}^{\pi} \rightarrow k \rangle$  **and**  $\langle \forall k. \neg j \text{ cd}^{\pi} \rightarrow k \rangle$  **shows**  $\langle i = j \rangle$   
**proof** (*rule ccontr,cases*)  
  **assume**  $\langle i < j \rangle$  **thus**  $\langle \text{False} \rangle$  **using** *assms claim''* **by** *blast*

next

assume  $\langle \neg i < j \rangle \langle i \neq j \rangle$   
 hence  $\langle j < i \rangle$  by *auto*  
 thus  $\langle \text{False} \rangle$  using *assms claim''* by (*metis loop-has-cd'*)

qed

lemma *icd-no-cd-path-shift*: assumes  $\langle i \text{ icd}^\pi \rightarrow 0 \rangle$  shows  $\langle (\forall k. \neg i - 1 \text{ cd}^{\pi \ll 1} \rightarrow k) \rangle$

proof (*rule,rule ccontr,goal-cases*)

case ( $1 k$ )  
 hence \*:  $\langle i - 1 \text{ cd}^{\pi \ll 1} \rightarrow k \rangle$  by *simp*  
 have \*\*:  $\langle 1 \leq k + 1 \rangle$  by *simp*  
 have \*\*\*:  $\langle \text{is-path } \pi \rangle$  by (*metis assms is-icdi-def*)  
 hence  $\langle i \text{ cd}^\pi \rightarrow k+1 \rangle$  using *cd-path-shift[OF \*\* \*\*\*]* \* by *auto*  
 moreover  
 hence  $\langle k+1 < i \rangle$  unfolding *is-cdi-def* by *simp*  
 moreover  
 have  $\langle 0 < k + 1 \rangle$  by *simp*  
 ultimately show  $\langle \text{False} \rangle$  using *assms[unfolded is-icdi-def]* by *auto*

qed

lemma *claim'*: assumes *path* $\pi$ :  $\langle \text{is-path } \pi \rangle$  and *path* $\pi'$ :  $\langle \text{is-path } \pi' \rangle$  and

$\pi i$ :  $\langle \pi i = \pi' i' \rangle$  and  $\pi j$ :  $\langle \pi j = \pi' j' \rangle$  and *not-cd*:

$\langle i \text{ icd}^\pi \rightarrow 0 \rangle \langle j \text{ icd}^\pi \rightarrow 0 \rangle$

$\langle i' \text{ icd}^{\pi'} \rightarrow 0 \rangle \langle j' \text{ icd}^{\pi'} \rightarrow 0 \rangle$

and *ilj*:  $\langle i < j \rangle$

and *nret*:  $\langle \pi i \neq \text{return} \rangle$

shows  $\langle i' < j' \rangle$

proof –

have *g0*:  $\langle 0 < i \rangle \langle 0 < j \rangle \langle 0 < i' \rangle \langle 0 < j' \rangle$  using *not-cd[unfolded is-icdi-def is-cdi-def]* by *auto*

have  $\langle (\pi \ll 1) (i - 1) = (\pi' \ll 1) (i' - 1) \rangle \langle (\pi \ll 1) (j - 1) = (\pi' \ll 1) (j' - 1) \rangle$  using  $\pi i \pi j g0$  by *auto*

moreover

have  $\langle \forall k. \neg (j - 1) \text{ cd}^{\pi \ll 1} \rightarrow k \rangle \langle \forall k. \neg (i' - 1) \text{ cd}^{\pi' \ll 1} \rightarrow k \rangle$

by (*metis icd-no-cd-path-shift not-cd(2)*) (*metis icd-no-cd-path-shift not-cd(3)*)

moreover

have  $\langle \text{is-path } (\pi \ll 1) \rangle \langle \text{is-path } (\pi' \ll 1) \rangle$  using *path* $\pi$  *path* $\pi'$  *path-path-shift* by *blast+*

moreover

have  $\langle (\pi \ll 1) (i - 1) \neq \text{return} \rangle$  using *g0 nret* by *auto*

moreover

have  $\langle i - 1 < j - 1 \rangle$  using *g0 ilj* by *auto*

ultimately have  $\langle i' - 1 < j' - 1 \rangle$  using *claim''* by *blast*

thus  $\langle i' < j' \rangle$  by *auto*

qed

lemma *other-claim*: assumes *path*:  $\langle \text{is-path } \pi \rangle$  and *eq*:  $\langle \pi i = \pi j \rangle$  and  $\langle \pi i \neq \text{return} \rangle$

and *icd*:  $\langle i \text{ icd}^\pi \rightarrow 0 \rangle$  and  $\langle j \text{ icd}^\pi \rightarrow 0 \rangle$  shows  $\langle i = j \rangle$  proof (*rule ccontr,cases*)

assume  $\langle i < j \rangle$  thus  $\langle \text{False} \rangle$  using *assms claim'* by *blast*

next

assume  $\langle \neg i < j \rangle \langle i \neq j \rangle$

hence  $\langle j < i \rangle$  by *auto*

thus  $\langle \text{False} \rangle$  using *assms claim'* by (*metis less-not-refl*)

qed

lemma *cd-trans0*: assumes  $\langle j \text{ cd}^\pi \rightarrow 0 \rangle$  and  $\langle k \text{ cd}^\pi \rightarrow j \rangle$  shows  $\langle k \text{ cd}^\pi \rightarrow 0 \rangle$  proof (*rule ccontr*)

have *path*:  $\langle \text{is-path } \pi \rangle$  and *ij*:  $\langle 0 < j \rangle$  and *jk*:  $\langle j < k \rangle$

and *nret*:  $\langle \pi j \neq \text{return} \rangle \langle \pi k \neq \text{return} \rangle$

and *noipdi*:  $\langle \forall l \in \{0..j\}. \pi l \neq \text{ipd } (\pi 0) \rangle$

**and**  $\text{noipdj}: \langle \forall l \in \{j..k\}. \pi l \neq \text{ipd}(\pi j) \rangle$   
**using** *assms* **unfolding** *is-cdi-def* **by** *auto*  
**assume**  $\langle \neg k \text{ cd}^\pi \rightarrow 0 \rangle$   
**hence**  $\langle \exists l \in \{0..k\}. \pi l = \text{ipd}(\pi 0) \rangle$  **unfolding** *is-cdi-def* **using** *path ij jk nret* **by** *force*  
**then obtain**  $l$  **where**  $\langle l \in \{0..k\} \rangle$  **and**  $l: \langle \pi l = \text{ipd}(\pi 0) \rangle$  **by** *auto*  
**hence**  $jl: \langle j < l \rangle$  **and**  $lk: \langle l \leq k \rangle$  **using** *noipdi ij* **by** *auto*  
**have**  $pdj: \langle \text{ipd}(\pi 0) \text{ pd} \rightarrow \pi j \rangle$  **proof** (*rule ccontr*)  
  **have**  $\langle \pi j \in \text{nodes} \rangle$  **using** *path* **by** (*metis path-nodes*)  
  **moreover**  
  **assume**  $\langle \neg \text{ipd}(\pi 0) \text{ pd} \rightarrow \pi j \rangle$   
  **ultimately**  
  **obtain**  $\pi' n$  **where**  $*$ :  $\langle \text{is-path } \pi' \rangle \langle \pi' 0 = \pi j \rangle \langle \pi' n = \text{return} \rangle \langle \forall k \leq n. \pi' k \neq \text{ipd}(\pi 0) \rangle$  **using** *no-pd-path*  
**by** *metis*  
  **hence**  $\text{path}'$ :  $\langle \text{is-path}(\pi @^j \pi') \rangle$  **by** (*metis path path-cons*)  
  **moreover**  
  **have**  $\langle \forall k \leq j + n. (\pi @^j \pi') k \neq \text{ipd}(\pi 0) \rangle$  **using** *noipdi \*(4)* **by** *auto*  
  **moreover**  
  **have**  $\langle (\pi @^j \pi') 0 = \pi 0 \rangle$  **by** *auto*  
  **moreover**  
  **have**  $\langle (\pi @^j \pi')(j + n) = \text{return} \rangle$  **using** *\*(2,3)* **by** *auto*  
  **ultimately**  
  **have**  $\langle \neg \text{ipd}(\pi 0) \text{ pd} \rightarrow \pi 0 \rangle$  **unfolding** *is-pd-def* **by** *metis*  
  **thus**  $\langle \text{False} \rangle$  **by** (*metis is-ipd-def ij ipd-is-ipd nret(1) path path-nodes term-path-stable less-imp-le*)  
**qed**  
**hence**  $\langle (\pi \ll j)(l-j) \text{ pd} \rightarrow (\pi \ll j) 0 \rangle$  **using** *jl l* **by** *auto*  
**moreover**  
**have**  $\langle \text{is-path}(\pi \ll j) \rangle$  **by** (*metis path path-path-shift*)  
**moreover**  
**have**  $\langle \pi l \neq \text{return} \rangle$  **by** (*metis lk nret(2) path term-path-stable*)  
**hence**  $\langle (\pi \ll j)(l-j) \neq \text{return} \rangle$  **using** *jl* **by** *auto*  
**moreover**  
**have**  $\langle \pi j \neq \text{ipd}(\pi 0) \rangle$  **using** *noipdi* **by** *force*  
**hence**  $\langle (\pi \ll j)(l-j) \neq (\pi \ll j) 0 \rangle$  **using** *jl l* **by** *auto*  
**ultimately**  
**obtain**  $k'$  **where**  $\langle k' \leq l-j \rangle$  **and**  $\langle (\pi \ll j) k' = \text{ipd}((\pi \ll j) 0) \rangle$  **using** *path-pd-ipd0'* **by** *blast*  
**hence**  $\langle j + k' \in \{j..k\} \rangle \langle \pi(j+k') = \text{ipd}(\pi j) \rangle$  **using** *jl lk* **by** *auto*  
**thus**  $\langle \text{False} \rangle$  **using** *noipdj* **by** *auto*  
**qed**

**lemma** *cd-trans*: **assumes**  $\langle j \text{ cd}^\pi \rightarrow i \rangle$  **and**  $\langle k \text{ cd}^\pi \rightarrow j \rangle$  **shows**  $\langle k \text{ cd}^\pi \rightarrow i \rangle$  **proof** –  
  **have**  $\text{path}$ :  $\langle \text{is-path } \pi \rangle$  **using** *assms is-cdi-def* **by** *auto*  
  **have**  $ij$ :  $\langle i < j \rangle$  **using** *assms is-cdi-def* **by** *auto*  
  **let**  $\langle ?\pi \rangle = \langle \pi \ll i \rangle$   
  **have**  $\langle j-i \text{ cd}^{?\pi} \rightarrow 0 \rangle$  **using** *assms(1) cd-path-shift0 path* **by** *auto*  
  **moreover**  
  **have**  $\langle k-i \text{ cd}^{?\pi} \rightarrow j-i \rangle$  **by** (*metis assms(2) cd-path-shift is-cdi-def ij less-imp-le-nat*)  
  **ultimately**  
  **have**  $\langle k-i \text{ cd}^{?\pi} \rightarrow 0 \rangle$  **using** *cd-trans0* **by** *auto*  
  **thus**  $\langle k \text{ cd}^\pi \rightarrow i \rangle$  **using** *path cd-path-shift0* **by** *auto*  
**qed**

**lemma** *excd-impl-eried*: **assumes**  $\langle \exists k. i \text{ cd}^\pi \rightarrow k \rangle$  **shows**  $\langle \exists k. i \text{ icd}^\pi \rightarrow k \rangle$   
**using** *assms* **proof**(*induction <i> arbitrary: <pi> rule: less-induct*)  
  **case** (*less i*)  
  **then obtain**  $k$  **where**  $k: \langle i \text{ cd}^\pi \rightarrow k \rangle$  **by** *auto*  
  **hence**  $ip$ :  $\langle \text{is-path } \pi \rangle$  **unfolding** *is-cdi-def* **by** *auto*

**show**  $\langle ?case \rangle$  **proof** (*cases*)  
**assume**  $*$ :  $\langle \forall m \in \{k < .. < i\}. \neg i \text{ cd}^\pi \rightarrow m \rangle$   
**hence**  $\langle i \text{ icd}^\pi \rightarrow k \rangle$  **using**  $k \text{ ip}$  **unfolding** *is-icdi-def* **by** *auto*  
**thus**  $\langle ?case \rangle$  **by** *auto*  
**next**  
**assume**  $\langle \neg (\forall m \in \{k < .. < i\}. \neg i \text{ cd}^\pi \rightarrow m) \rangle$   
**then obtain**  $m$  **where**  $m$ :  $\langle m \in \{k < .. < i\} \langle i \text{ cd}^\pi \rightarrow m \rangle$  **by** *blast*  
**hence**  $\langle i - m \text{ cd}^{\pi \ll m} \rightarrow 0 \rangle$  **by** (*metis cd-path-shift0 is-cdi-def*)  
**moreover**  
**have**  $\langle i - m < i \rangle$  **using**  $m$  **by** *auto*  
**ultimately**  
**obtain**  $k'$  **where**  $k'$ :  $\langle i - m \text{ icd}^{\pi \ll m} \rightarrow k' \rangle$  **using** *less(1)* **by** *blast*  
**hence**  $\langle i \text{ icd}^\pi \rightarrow k' + m \rangle$  **using** *ip*  
**by** (*metis add commute add-diff-cancel-right' icd-path-shift le-add1*)  
**thus**  $\langle ?case \rangle$  **by** *auto*  
**qed**  
**qed**

**lemma** *cd-split*: **assumes**  $\langle i \text{ cd}^\pi \rightarrow k \rangle$  **and**  $\langle \neg i \text{ icd}^\pi \rightarrow k \rangle$  **obtains**  $m$  **where**  $\langle i \text{ icd}^\pi \rightarrow m \rangle$  **and**  $\langle m \text{ cd}^\pi \rightarrow k \rangle$   
**proof** –  
**have**  $ki$ :  $\langle k < i \rangle$  **using** *assms is-cdi-def* **by** *auto*  
**obtain**  $m$  **where**  $m$ :  $\langle i \text{ icd}^\pi \rightarrow m \rangle$  **using** *assms(1)* **by** (*metis excd-impl-excd*)  
**hence**  $\langle k \leq m \rangle$  **unfolding** *is-icdi-def* **using**  $ki$  *assms(1)* **by** *force*  
**hence**  $km$ :  $\langle k < m \rangle$  **using**  $m$  *assms(2)* **by** (*metis le-eq-less-or-eq*)  
**moreover have**  $\langle \pi m \neq \text{return} \rangle$  **using**  $m$  **unfolding** *is-icdi-def is-cdi-def* **by** (*simp, metis term-path-stable less-imp-le*)  
**moreover have**  $\langle m < i \rangle$  **using**  $m$  **unfolding** *is-cdi-def is-icdi-def* **by** *auto*  
**ultimately**  
**have**  $\langle m \text{ cd}^\pi \rightarrow k \rangle$  **using** *assms(1)* **unfolding** *is-cdi-def* **by** *auto*  
**with**  $m$  **that show**  $\langle \text{thesis} \rangle$  **by** *auto*  
**qed**

**lemma** *cd-induct*[*consumes 1, case-names base IS*]: **assumes** *prem*:  $\langle i \text{ cd}^\pi \rightarrow k \rangle$  **and** *base*:  $\langle \bigwedge i. i \text{ icd}^\pi \rightarrow k \implies P i \rangle$   
**and** *IH*:  $\langle \bigwedge k' i'. k' \text{ cd}^\pi \rightarrow k \implies P k' \implies i' \text{ icd}^\pi \rightarrow k' \implies P i' \rangle$  **shows**  $\langle P i \rangle$   
**using** *prem IH* **proof** (*induction*  $\langle i \rangle$  *rule: less-induct,cases*)  
**case** (*less i*)  
**assume**  $\langle i \text{ icd}^\pi \rightarrow k \rangle$   
**thus**  $\langle P i \rangle$  **using** *base* **by** *simp*  
**next**  
**case** (*less i'*)  
**assume**  $\langle \neg i' \text{ icd}^\pi \rightarrow k \rangle$   
**then obtain**  $k'$  **where**  $k'$ :  $\langle i' \text{ icd}^\pi \rightarrow k' \rangle \langle k' \text{ cd}^\pi \rightarrow k \rangle$  **using** *less cd-split* **by** *blast*  
**hence**  $icdk$ :  $\langle i' \text{ cd}^\pi \rightarrow k' \rangle$  **using** *is-icdi-def* **by** *auto*  
**note**  $ih=less(3)[OF k'(2) - k'(1)]$   
**have**  $ki$ :  $\langle k' < i' \rangle$  **using**  $k'$  *is-icdi-def is-cdi-def* **by** *auto*  
**have**  $\langle P k' \rangle$  **using** *less(1)[OF ki k'(2)] less(3)* **by** *auto*  
**thus**  $\langle P i' \rangle$  **using** *ih* **by** *simp*  
**qed**

**lemma** *cdi-prefix*:  $\langle n \text{ cd}^\pi \rightarrow m \implies m < n' \implies n' \leq n \implies n' \text{ cd}^\pi \rightarrow m \rangle$  **unfolding** *is-cdi-def*  
**by** (*simp, metis term-path-stable*)

**lemma** *cr-wn'*: **assumes**  $1$ :  $\langle n \text{ cd}^\pi \rightarrow m \rangle$  **and**  $nc$ :  $\langle \neg m' \text{ cd}^\pi \rightarrow m \rangle$  **and**  $3$ :  $\langle m < m' \rangle$  **shows**  $\langle n < m' \rangle$   
**proof** (*rule ccontr*)  
**assume**  $\langle \neg n < m' \rangle$

hence  $\langle m' \leq n \rangle$  by *simp*  
hence  $\langle m' \text{ cd}^\pi \rightarrow m \rangle$  by (*metis 1 3 cdi-prefix*)  
thus  $\langle \text{False} \rangle$  using *nc* by *simp*  
qed

lemma *cr-wn''*: assumes  $\langle i \text{ cd}^\pi \rightarrow m \rangle$  and  $\langle j \text{ cd}^\pi \rightarrow n \rangle$  and  $\langle \neg m \text{ cd}^\pi \rightarrow n \rangle$  and  $\langle i \leq j \rangle$  shows  $\langle m \leq n \rangle$   
**proof** (*rule ccontr*)  
assume  $\langle \neg m \leq n \rangle$   
hence *nm*:  $\langle n < m \rangle$  by *auto*  
**moreover**  
have  $\langle m < j \rangle$  using *assms(1) assms(4) unfolding is-cdi-def* by *auto*  
**ultimately**  
have  $\langle m \text{ cd}^\pi \rightarrow n \rangle$  using *assms(2) cdi-prefix* by *auto*  
thus  $\langle \text{False} \rangle$  using *assms(3)* by *auto*  
qed

lemma *ret-no-cd*: assumes  $\langle \pi n = \text{return} \rangle$  shows  $\langle \neg n \text{ cd}^\pi \rightarrow k \rangle$  by (*metis assms is-cdi-def*)

lemma *ipd-not-self*: assumes  $\langle x \in \text{nodes} \rangle \langle x \neq \text{return} \rangle$  shows  $\langle x \neq \text{ipd } x \rangle$  by (*metis is-ipd-def assms ipd-is-ipd*)

lemma *icd-cs*: assumes  $\langle l \text{ icd}^\pi \rightarrow k \rangle$  shows  $\langle \text{cs}^\pi l = \text{cs}^\pi k @ [\pi l] \rangle$   
**proof** –  
from *assms* have  $\langle k = (\text{THE } k. l \text{ icd}^\pi \rightarrow k) \rangle$  by (*metis icd-is-the-icd*)  
with *assms* show  $\langle ?thesis \rangle$  by *auto*  
qed

lemma *cd-not-pd*: assumes  $\langle l \text{ cd}^\pi \rightarrow k \rangle \langle \pi l \neq \pi k \rangle$  shows  $\langle \neg \pi l \text{ pd} \rightarrow \pi k \rangle$  **proof**  
assume *pd*:  $\langle \pi l \text{ pd} \rightarrow \pi k \rangle$   
have *nret*:  $\langle \pi k \neq \text{return} \rangle$  by (*metis assms(1) pd pd-ret-is-ret ret-no-cd*)  
have *kl*:  $\langle k < l \rangle$  by (*metis is-cdi-def assms(1)*)  
have *path*:  $\langle \text{is-path } \pi \rangle$  by (*metis is-cdi-def assms(1)*)  
from *path-pd-ipd*[*OF path nret assms(2) pd kl*]  
obtain *n* where  $\langle k < n \rangle \langle n \leq l \rangle \langle \pi n = \text{ipd } (\pi k) \rangle$ .  
thus  $\langle \text{False} \rangle$  using *assms(1) unfolding is-cdi-def* by *auto*  
qed

lemma *cd-ipd-is-cd*: assumes  $\langle k < m \rangle \langle \pi m = \text{ipd } (\pi k) \rangle \langle \forall n \in \{k..<m\}. \pi n \neq \text{ipd } (\pi k) \rangle$  and *mcj*:  $\langle m \text{ cd}^\pi \rightarrow j \rangle$  shows  $\langle k \text{ cd}^\pi \rightarrow j \rangle$  **proof** *cases*  
assume  $\langle j < k \rangle$  thus  $\langle k \text{ cd}^\pi \rightarrow j \rangle$  by (*metis mcj assms(1) cdi-prefix less-imp-le-nat*)  
**next**  
assume  $\langle \neg j < k \rangle$   
hence *kj*:  $\langle k \leq j \rangle$  by *simp*  
have  $\langle k < j \rangle$  **apply** (*rule ccontr*) using *kj assms mcj* by (*auto, metis is-cdi-def is-ipd-def cd-not-pd ipd-is-ipd path-nodes term-path-stable less-imp-le*)  
**moreover**  
have  $\langle j < m \rangle$  using *mcj is-cdi-def* by *auto*  
hence  $\langle \forall n \in \{k..j\}. \pi n \neq \text{ipd } (\pi k) \rangle$  using *assms(3)* by *force*  
**ultimately**  
have  $\langle j \text{ cd}^\pi \rightarrow k \rangle$  by (*metis mcj is-cdi-def term-path-stable less-imp-le*)  
hence  $\langle m \text{ cd}^\pi \rightarrow k \rangle$  by (*metis mcj cd-trans*)  
hence  $\langle \text{False} \rangle$  by (*metis is-cdi-def is-ipd-def assms(2) cd-not-pd ipd-is-ipd path-nodes term-path-stable less-imp-le*)  
thus  $\langle ?thesis \rangle$  by *simp*  
qed

lemma *ipd-pd-cd0*: assumes *lcd*:  $\langle n \text{ cd}^\pi \rightarrow 0 \rangle$  shows  $\langle \text{ipd } (\pi 0) \text{ pd} \rightarrow (\pi n) \rangle$   
**proof** –

**obtain**  $k\ l$  **where**  $\pi 0: \langle \pi\ 0 = k \rangle$  **and**  $\pi n: \langle \pi\ n = l \rangle$  **and**  $cdi: \langle n\ cd^\pi \rightarrow 0 \rangle$  **using** *lcd unfolding is-cdi-def*  
**by** *blast*  
**have**  $nret: \langle k \neq return \rangle$  **by** (*metis is-cdi-def  $\pi 0\ cdi\ term\ path\ stable\ less\ imp\ le$* )  
**have**  $path: \langle is\ path\ \pi \rangle$  **and**  $ipd: \langle \forall\ i \leq n. \pi\ i \neq ipd\ k \rangle$  **using** *cdi unfolding is-cdi-def  $\pi 0$*  **by** *auto*  
**{**  
  **fix**  $\pi'\ n'$   
  **assume**  $path': \langle is\ path\ \pi' \rangle$   
  **and**  $\pi' 0: \langle \pi'\ 0 = l \rangle$   
  **and**  $ret: \langle \pi'\ n' = return \rangle$   
  **have**  $\langle is\ path\ (\pi\ @^n\ \pi') \rangle$  **using** *path path'  $\pi n\ \pi' 0$*  **by** (*metis path-cons*)  
  **moreover**  
  **have**  $\langle (\pi\ @^n\ \pi')\ (n+n') = return \rangle$  **using** *ret  $\pi n\ \pi' 0$*  **by** *auto*  
  **moreover**  
  **have**  $\langle (\pi\ @^n\ \pi')\ 0 = k \rangle$  **using**  $\pi 0$  **by** *auto*  
  **moreover**  
  **have**  $\langle ipd\ k\ pd \rightarrow k \rangle$  **by** (*metis is-ipd-def path  $\pi 0\ ipd\ is\ ipd\ nret\ path\ nodes$* )  
  **ultimately**  
  **obtain**  $k'$  **where**  $k': \langle k' \leq n+n' \rangle$   $\langle (\pi\ @^n\ \pi')\ k' = ipd\ k \rangle$  **by** (*metis pd-intro*)  
  **have**  $\langle \neg\ k' \leq n \rangle$  **proof**  
    **assume**  $\langle k' \leq n \rangle$   
    **hence**  $\langle (\pi\ @^n\ \pi')\ k' = \pi\ k' \rangle$  **by** *auto*  
    **thus**  $\langle False \rangle$  **using**  $k'(2)\ ipd$  **by** (*metis  $\langle k' \leq n \rangle$* )  
  **qed**  
  **hence**  $\langle (\pi\ @^n\ \pi')\ k' = \pi'\ (k' - n) \rangle$  **by** *auto*  
  **moreover**  
  **have**  $\langle (k' - n) \leq n' \rangle$  **using**  $k'$  **by** *simp*  
  **ultimately**  
  **have**  $\langle \exists\ k' \leq n'. \pi'\ k' = ipd\ k \rangle$  **unfolding**  $k'$  **by** *auto*  
**}**  
  **moreover**  
  **have**  $\langle l \in nodes \rangle$  **by** (*metis  $\pi n\ path\ path\ nodes$* )  
  **ultimately show**  $\langle ipd\ (\pi\ 0)\ pd \rightarrow (\pi\ n) \rangle$  **unfolding** *is-pd-def* **by** (*simp add:  $\pi 0\ \pi n$* )  
**qed**

**lemma** *ipd-pd-cd*: **assumes**  $lcd: \langle l\ cd^\pi \rightarrow k \rangle$  **shows**  $\langle ipd\ (\pi\ k)\ pd \rightarrow (\pi\ l) \rangle$   
**proof** –

**have**  $\langle l - k\ cd^{\pi \ll k} \rightarrow 0 \rangle$  **using** *lcd cd-path-shift0 is-cdi-def* **by** *blast*  
  **moreover**  
  **note** *ipd-pd-cd0[OF this]*  
  **moreover**  
  **have**  $\langle (\pi\ \ll\ k)\ 0 = \pi\ k \rangle$  **by** *auto*  
  **moreover**  
  **have**  $\langle k < l \rangle$  **using** *lcd unfolding is-cdi-def* **by** *simp*  
  **then have**  $\langle (\pi\ \ll\ k)\ (l - k) = \pi\ l \rangle$  **by** *simp*  
  **ultimately show**  $\langle ?thesis \rangle$  **by** *simp*  
**qed**

**lemma** *cd-is-cd-ipd*: **assumes**  $km: \langle k < m \rangle$  **and**  $ipd: \langle \pi\ m = ipd\ (\pi\ k) \rangle$   $\langle \forall\ n \in \{k..<m\}. \pi\ n \neq ipd\ (\pi\ k) \rangle$  **and**  
 $cdj: \langle k\ cd^\pi \rightarrow j \rangle$  **and**  $nipdj: \langle ipd\ (\pi\ j) \neq \pi\ m \rangle$  **shows**  $\langle m\ cd^\pi \rightarrow j \rangle$  **proof** –

**have**  $path: \langle is\ path\ \pi \rangle$   
  **and**  $jk: \langle j < k \rangle$   
  **and**  $nretj: \langle \pi\ k \neq return \rangle$   
  **and**  $nipd: \langle \forall\ l \in \{j..k\}. \pi\ l \neq ipd\ (\pi\ j) \rangle$  **using** *cdj is-cdi-def* **by** *auto*  
  **have**  $pd: \langle ipd\ (\pi\ j)\ pd \rightarrow \pi\ m \rangle$  **by** (*metis atLeastAtMost-iff cdj ipd(1) ipd-pd-cd jk le-refl less-imp-le nipd nretj path path-nodes pd-pd-ipd*)  
  **have**  $nretm: \langle \pi\ m \neq return \rangle$  **by** (*metis nipdj pd pd-ret-is-ret*)



**have**  $jm$ :  $\langle j < m \rangle$  **using**  $jk\ km$  **by** *simp*  
**show**  $\langle m\ cd^\pi \rightarrow j \rangle$  **proof** (*rule ccontr*)  
  **assume**  $ncdj$ :  $\langle \neg\ m\ cd^\pi \rightarrow j \rangle$   
  **hence**  $\langle \exists\ l \in \{j..m\}. \pi\ l = ipd\ (\pi\ j) \rangle$  **unfolding** *is-cdi-def* **by** (*metis jm nretm path*)  
  **then obtain**  $l$   
  **where**  $jl$ :  $\langle j \leq l \rangle$  **and**  $\langle l \leq m \rangle$   
  **and**  $lipd$ :  $\langle \pi\ l = ipd\ (\pi\ j) \rangle$  **by** *force*  
  **hence**  $lm$ :  $\langle l < m \rangle$  **using**  $nipdj$  **by** (*metis le-eq-less-or-eq*)  
  **have**  $npd$ :  $\langle \neg\ ipd\ (\pi\ k)\ pd \rightarrow \pi\ l \rangle$  **by** (*metis ipd(1) lipd nipdj pd pd-antisym*)  
  **have**  $nd$ :  $\langle \pi\ l \in nodes \rangle$  **using** *path path-nodes* **by** *simp*  
  **from** *no-pd-path[OF nd npd]*  
  **obtain**  $\pi'\ n$  **where**  $path'$ :  $\langle is-path\ \pi' \rangle$  **and**  $\pi'\ 0$ :  $\langle \pi'\ 0 = \pi\ l \rangle$  **and**  $\pi'\ n$ :  $\langle \pi'\ n = return \rangle$  **and**  $nipd$ :  $\langle \forall\ ka \leq n.$   
 $\pi'\ ka \neq ipd\ (\pi\ k) \rangle$  .  
  **let**  $\langle ?\pi \rangle = \langle \pi @^l\ \pi' \rangle \ll k$   
  **have**  $path''$ :  $\langle is-path\ ?\pi \rangle$  **by** (*metis \pi'0 path path' path-cons path-path-shift*)  
  **moreover**  
  **have**  $kl$ :  $\langle k < l \rangle$  **using**  $lipd\ cdj\ jl$  **unfolding** *is-cdi-def* **by** *fastforce*  
  **have**  $\langle ?\pi\ 0 = \pi\ k \rangle$  **using**  $kl$  **by** *auto*  
  **moreover**  
  **have**  $\langle ?\pi\ (l + n - k) = return \rangle$  **using**  $\pi'\ n\ \pi'\ 0\ kl$  **by** *auto*  
  **moreover**  
  **have**  $\langle ipd\ (\pi\ k)\ pd \rightarrow \pi\ k \rangle$  **by** (*metis is-ipd-def ipd-is-ipd nretj path path-nodes*)  
  **ultimately**  
  **obtain**  $l'$  **where**  $l'$ :  $\langle l' \leq (l + n - k) \rangle$   $\langle ?\pi\ l' = ipd\ (\pi\ k) \rangle$  **unfolding** *is-pd-def* **by** *blast*  
  **show**  $\langle False \rangle$  **proof** (*cases*)  
  **assume**  $*$ :  $\langle k + l' \leq l \rangle$   
  **hence**  $\langle \pi\ (k + l') = ipd\ (\pi\ k) \rangle$  **using**  $l'$  **by** *auto*  
  **moreover**  
  **have**  $\langle k + l' < m \rangle$  **by** (*metis \* dual-order.strict-trans2 lm*)  
  **ultimately**  
  **show**  $\langle False \rangle$  **using**  $ipd(2)$  **by** *simp*  
**next**  
  **assume**  $\langle \neg\ k + l' \leq l \rangle$   
  **hence**  $\langle \pi'\ (k + l' - l) = ipd\ (\pi\ k) \rangle$  **using**  $l'$  **by** *auto*  
  **moreover**  
  **have**  $\langle k + l' - l \leq n \rangle$  **using**  $l'\ kl$  **by** *linarith*  
  **ultimately**  
  **show**  $\langle False \rangle$  **using**  $nipd$  **by** *auto*  
  **qed**  
**qed**  
**qed**

**lemma** *ipd-icd-greatest-cd-not-ipd*: **assumes**  $ipd$ :  $\langle \pi\ m = ipd\ (\pi\ k) \rangle$   $\langle \forall\ n \in \{k..<m\}. \pi\ n \neq ipd\ (\pi\ k) \rangle$   
**and**  $km$ :  $\langle k < m \rangle$  **and**  $icdj$ :  $\langle m\ icd^\pi \rightarrow j \rangle$  **shows**  $\langle j = (GREATEST\ j. k\ cd^\pi \rightarrow j \wedge ipd\ (\pi\ j) \neq \pi\ m) \rangle$   
**proof** –

**let**  $\langle ?j \rangle = \langle GREATEST\ j. k\ cd^\pi \rightarrow j \wedge ipd\ (\pi\ j) \neq \pi\ m \rangle$   
  **have**  $kcdj$ :  $\langle k\ cd^\pi \rightarrow j \rangle$  **using** *assms cd-ipd-is-cd is-icdi-def* **by** *blast*  
  **have**  $nipd$ :  $\langle ipd\ (\pi\ j) \neq \pi\ m \rangle$  **using**  $icdj$  **unfolding** *is-icdi-def is-cdi-def* **by** *auto*  
  **have**  $bound$ :  $\langle \bigwedge\ j. k\ cd^\pi \rightarrow j \wedge ipd\ (\pi\ j) \neq \pi\ m \implies j \leq k \rangle$  **unfolding** *is-cdi-def* **by** *simp*  
  **have**  $exists$ :  $\langle k\ cd^\pi \rightarrow j \wedge ipd\ (\pi\ j) \neq \pi\ m \rangle$  (**is**  $\langle ?P\ j \rangle$ ) **using**  $kcdj\ nipd$  **by** *auto*  
  **note** *GreatestI-nat[of \langle ?P \rangle - \langle k \rangle, OF exists] Greatest-le-nat[of \langle ?P \rangle \langle j \rangle \langle k \rangle, OF exists]*  
  **hence**  $kcdj'$ :  $\langle k\ cd^\pi \rightarrow ?j \rangle$  **and**  $ipd'$ :  $\langle ipd\ (\pi\ ?j) \neq \pi\ m \rangle$  **and**  $jj$ :  $\langle j \leq ?j \rangle$  **using**  $bound$  **by** *auto*  
  **hence**  $medj'$ :  $\langle m\ cd^\pi \rightarrow ?j \rangle$  **using**  $ipd\ km\ cd-is-cd-ipd$  **by** *auto*  
  **show**  $\langle j = ?j \rangle$  **proof** (*rule ccontr*)  
  **assume**  $\langle j \neq ?j \rangle$   
  **hence**  $jlj$ :  $\langle j < ?j \rangle$  **using**  $jj$  **by** *simp*

**moreover**  
**have**  $\langle ?j < m \rangle$  **using**  $kcdj'$   $km$  **unfolding**  $is-cdi-def$  **by**  $auto$   
**ultimately**  
**show**  $\langle False \rangle$  **using**  $icdj$   $mcdj'$  **unfolding**  $is-icdi-def$  **by**  $auto$   
**qed**  
**qed**

**lemma**  $cd-impl-icd-cd$ : **assumes**  $\langle i cd^\pi \rightarrow l \rangle$  **and**  $\langle i icd^\pi \rightarrow k \rangle$  **and**  $\langle \neg i icd^\pi \rightarrow l \rangle$  **shows**  $\langle k cd^\pi \rightarrow l \rangle$   
**using**  $assms$   $cd-split$   $icd-uniq$  **by**  $metis$

**lemma**  $cdi-is-cd-icdi$ : **assumes**  $\langle k icd^\pi \rightarrow j \rangle$  **shows**  $\langle k cd^\pi \rightarrow i \iff j cd^\pi \rightarrow i \vee i = j \rangle$   
**by** ( $metis$   $assms$   $cd-impl-icd-cd$   $cd-trans$   $icd-imp-cd$   $icd-uniq$ )

**lemma**  $same-ipd-stable$ : **assumes**  $\langle k cd^\pi \rightarrow i \rangle$   $\langle k cd^\pi \rightarrow j \rangle$   $\langle i < j \rangle$   $\langle ipd(\pi i) = ipd(\pi k) \rangle$  **shows**  $\langle ipd(\pi j) = ipd(\pi k) \rangle$

**proof** –

**have**  $jcdi$ :  $\langle j cd^\pi \rightarrow i \rangle$  **by** ( $metis$   $is-cdi-def$   $assms(1,2,3)$   $cr-wn'$   $le-antisym$   $less-imp-le-nat$ )  
**have** 1:  $\langle ipd(\pi j) pd \rightarrow \pi k \rangle$  **by** ( $metis$   $assms(2)$   $ipd-pd-cd$ )  
**have** 2:  $\langle ipd(\pi k) pd \rightarrow \pi j \rangle$  **by** ( $metis$   $assms(4)$   $ipd-pd-cd$   $jcdi$ )  
**have** 3:  $\langle ipd(\pi k) pd \rightarrow (ipd(\pi j)) \rangle$  **by** ( $metis$  2  $IFC-def.is-cdi-def$   $assms(1,2,4)$   $atLeastAtMost-iff$   $jcdi$   $less-imp-le$   $pd-node2$   $pd-pd-ipd$ )  
**have** 4:  $\langle ipd(\pi j) pd \rightarrow (ipd(\pi k)) \rangle$  **by** ( $metis$  1 2  $IFC-def.is-ipd-def$   $assms(2)$   $cd-not-pd$   $ipd-is-ipd$   $jcdi$   $pd-node2$   $ret-no-cd$ )  
**show**  $\langle ?thesis \rangle$  **using** 3 4  $pd-antisym$  **by**  $simp$   
**qed**

**lemma**  $icd-pd-intermediate'$ : **assumes**  $icd$ :  $\langle i icd^\pi \rightarrow k \rangle$  **and**  $j$ :  $\langle k < j \rangle$   $\langle j < i \rangle$  **shows**  $\langle \pi i pd \rightarrow (\pi j) \rangle$   
**using**  $j$  **proof** ( $induction$   $\langle i - j \rangle$   $arbitrary$ :  $\langle j \rangle$   $rule$ :  $less-induct$ )

**case** ( $less\ j$ )

**have**  $\langle \neg i cd^\pi \rightarrow j \rangle$  **using**  $less.prem$ s  $icd$  **unfolding**  $is-icdi-def$  **by**  $force$

**moreover**

**have**  $\langle is-path\ \pi \rangle$  **using**  $icd$  **by** ( $metis$   $is-icdi-def$ )

**moreover**

**have**  $\langle \pi i \neq return \rangle$  **using**  $icd$  **by** ( $metis$   $is-icdi-def$   $ret-no-cd$ )

**ultimately**

**have**  $\langle \exists l. j \leq l \wedge l \leq i \wedge \pi l = ipd(\pi j) \rangle$  **unfolding**  $is-cdi-def$  **using**  $less.prem$ s **by**  $auto$

**then obtain**  $l$  **where**  $l$ :  $\langle j \leq l \rangle$   $\langle l \leq i \rangle$   $\langle \pi l = ipd(\pi j) \rangle$  **by**  $blast$

**hence**  $lpd$ :  $\langle \pi l pd \rightarrow (\pi j) \rangle$  **by** ( $metis$   $is-ipd-def$   $\langle \pi i \neq return \rangle$   $\langle is-path\ \pi \rangle$   $ipd-is-ipd$   $path-nodes$   $term-path-stable$ )

**show**  $\langle ?case \rangle$  **proof** ( $cases$ )

**assume**  $\langle l = i \rangle$

**thus**  $\langle ?case \rangle$  **using**  $lpd$  **by**  $auto$

**next**

**assume**  $\langle l \neq i \rangle$

**hence**  $\langle l < i \rangle$  **using**  $l$  **by**  $simp$

**moreover**

**have**  $\langle j \neq l \rangle$  **using**  $l$  **by** ( $metis$   $is-ipd-def$   $\langle \pi i \neq return \rangle$   $\langle is-path\ \pi \rangle$   $ipd-is-ipd$   $path-nodes$   $term-path-stable$ )

**hence**  $\langle j < l \rangle$  **using**  $l$  **by**  $simp$

**moreover**

**hence**  $\langle i - l < i - j \rangle$  **by** ( $metis$   $diff-less-mono2$   $less.prem$ s(2))

**moreover**

**have**  $\langle k < l \rangle$  **by** ( $metis$   $l(1)$   $less.prem$ s(1)  $linorder-neqE-nat$   $not-le$   $order.strict-trans$ )

**ultimately**

**have**  $\langle \pi i pd \rightarrow (\pi l) \rangle$  **using**  $less.hyps$  **by**  $auto$

**thus**  $\langle ?case \rangle$  **using**  $lpd$  **by** ( $metis$   $pd-trans$ )

**qed**

**qed**

**lemma** *icd-pd-intermediate*: **assumes** *icd*:  $\langle i \text{ icd}^\pi \rightarrow k \rangle$  **and** *j*:  $\langle k < j \rangle \langle j \leq i \rangle$  **shows**  $\langle \pi i \text{ pd} \rightarrow (\pi j) \rangle$   
**using** *assms icd-pd-intermediate* [OF *assms(1,2)*] **apply** (*cases*  $\langle j < i \rangle$ , *metis*) **by** (*metis is-icdi-def le-neq-trans path-nodes pd-refl*)

**lemma** *no-icd-pd*: **assumes** *path*:  $\langle \text{is-path } \pi \rangle$  **and** *noicd*:  $\langle \forall l \geq n. \neg k \text{ icd}^\pi \rightarrow l \rangle$  **and** *nk*:  $\langle n \leq k \rangle$  **shows**  $\langle \pi k \text{ pd} \rightarrow \pi n \rangle$

**proof** *cases*

**assume**  $\langle \pi k = \text{return} \rangle$  **thus**  $\langle ?thesis \rangle$  **by** (*metis path path-nodes return-pd*)

**next**

**assume** *nret*:  $\langle \pi k \neq \text{return} \rangle$

**have** *nocd*:  $\langle \bigwedge l. n \leq l \implies \neg k \text{ cd}^\pi \rightarrow l \rangle$  **proof**

**fix** *l* **assume** *kcd*:  $\langle k \text{ cd}^\pi \rightarrow l \rangle$  **and** *nl*:  $\langle n \leq l \rangle$

**hence**  $\langle (k - n) \text{ cd}^{\pi \ll n} \rightarrow (l - n) \rangle$  **using** *cd-path-shift* [OF *nl path*] **by** *simp*

**hence**  $\langle \exists l. (k - n) \text{ icd}^{\pi \ll n} \rightarrow l \rangle$  **using** *excd-impl-exicd* **by** *blast*

**then obtain** *l'* **where**  $k - n \text{ icd}^{\pi \ll n} \rightarrow l' ..$

**hence**  $\langle k \text{ icd}^\pi \rightarrow (l' + n) \rangle$  **using** *icd-path-shift* [of  $\langle n \rangle \langle l' + n \rangle \langle \pi \rangle \langle k \rangle$ ] *path* **by** *auto*

**thus**  $\langle \text{False} \rangle$  **using** *noicd* **by** *auto*

**qed**

**hence**  $\langle \bigwedge l. n \leq l \implies l < k \implies \exists j \in \{l..k\}. \pi j = \text{ipd} (\pi l) \rangle$  **using** *path nret unfolding is-cdi-def* **by** *auto*

**thus**  $\langle ?thesis \rangle$  **using** *nk proof* (*induction*  $\langle k - n \rangle$  *arbitrary*:  $\langle n \rangle$  *rule*: *less-induct,cases*)

**case** (*less n*)

**assume**  $\langle n = k \rangle$

**thus**  $\langle ?case \rangle$  **using** *pd-refl path path-nodes* **by** *auto*

**next**

**case** (*less n*)

**assume**  $\langle n \neq k \rangle$

**hence** *nk*:  $\langle n < k \rangle$  **using** *less(3)* **by** *auto*

**with** *less(2)* **obtain** *j* **where** *jnk*:  $\langle j \in \{n..k\} \rangle$  **and** *ipdj*:  $\langle \pi j = \text{ipd} (\pi n) \rangle$  **by** *blast*

**have** *nretn*:  $\langle \pi n \neq \text{return} \rangle$  **using** *nk nret term-path-stable path* **by** *auto*

**with** *ipd-is-ipd path path-nodes is-ipd-def ipdj*

**have** *jpdn*:  $\langle \pi j \text{ pd} \rightarrow \pi n \rangle$  **by** *auto*

**show**  $\langle ?case \rangle$  **proof** *cases*

**assume**  $\langle j = k \rangle$  **thus**  $\langle ?case \rangle$  **using** *jpdn* **by** *simp*

**next**

**assume**  $\langle j \neq k \rangle$

**hence** *jk*:  $\langle j < k \rangle$  **using** *jnk* **by** *auto*

**have**  $\langle j \neq n \rangle$  **using** *ipdj* **by** (*metis ipd-not-self nretn path path-nodes*)

**hence** *nj*:  $\langle n < j \rangle$  **using** *jnk* **by** *auto*

**have**  $\ast$ :  $\langle k - j < k - n \rangle$  **using** *jk nj* **by** *auto*

**with** *less(1)* [OF  $\ast$ ] *less(2)* *jk nj*

**have**  $\langle \pi k \text{ pd} \rightarrow \pi j \rangle$  **by** *auto*

**thus**  $\langle ?thesis \rangle$  **using** *jpdn pd-trans* **by** *metis*

**qed**

**qed**

**qed**

**lemma** *first-pd-no-cd*: **assumes** *path*:  $\langle \text{is-path } \pi \rangle$  **and** *pd*:  $\langle \pi n \text{ pd} \rightarrow \pi 0 \rangle$  **and** *first*:  $\langle \forall l < n. \pi l \neq \pi n \rangle$   
**shows**  $\langle \forall l. \neg n \text{ cd}^\pi \rightarrow l \rangle$

**proof** (*rule ccontr, goal-cases*)

**case** 1

**then obtain** *l* **where** *ncdl*:  $\langle n \text{ cd}^\pi \rightarrow l \rangle$  **by** *blast*

**hence** *ln*:  $\langle l < n \rangle$  **using** *is-cdi-def* **by** *auto*

**have**  $\langle \neg \pi \ n \ pd \rightarrow \pi \ l \rangle$  **using** *ncdl cd-not-pd* **by** (*metis ln first*)  
**then obtain**  $\pi' \ n'$  **where** *path'*:  $\langle is-path \ \pi' \rangle$  **and**  $\pi 0$ :  $\langle \pi' \ 0 = \pi \ l \rangle$  **and**  $\pi n$ :  $\langle \pi' \ n' = return \rangle$  **and** *not* $\pi n$ :  $\langle \forall j \leq n'. \ \pi' \ j \neq \pi \ n \rangle$  **unfolding** *is-pd-def* **using** *path path-nodes* **by** *auto*  
**let**  $\langle ?\pi \rangle = \langle \pi @^l \ \pi' \rangle$

**have**  $\langle is-path \ ?\pi \rangle$  **by** (*metis  $\pi 0$  path path' path-cons*)  
**moreover**  
**have**  $\langle ?\pi \ 0 = \pi \ 0 \rangle$  **by** *auto*  
**moreover**  
**have**  $\langle ?\pi \ (n' + l) = return \rangle$  **using**  $\pi 0 \ \pi n$  **by** *auto*  
**ultimately**  
**obtain**  $j$  **where**  $j$ :  $\langle j \leq n' + l \rangle$  **and**  $jn$ :  $\langle ?\pi \ j = \pi \ n \rangle$  **using** *pd unfolding is-pd-def* **by** *blast*  
**show**  $\langle False \rangle$  **proof** *cases*  
**assume**  $\langle j \leq l \rangle$  **thus**  $\langle False \rangle$  **using**  $jn$  *first ln* **by** *auto*  
**next**  
**assume**  $\langle \neg j \leq l \rangle$  **thus**  $\langle False \rangle$  **using**  $j \ jn$  *not $\pi n$*  **by** *auto*  
**qed**  
**qed**

**lemma** *first-pd-no-icd*: **assumes** *path*:  $\langle is-path \ \pi \rangle$  **and** *pd*:  $\langle \pi \ n \ pd \rightarrow \pi \ 0 \rangle$  **and** *first*:  $\langle \forall l < n. \ \pi \ l \neq \pi \ n \rangle$   
**shows**  $\langle \forall l. \ \neg \ n \ icd^\pi \rightarrow l \rangle$   
**by** (*metis first first-pd-no-cd icd-imp-cd path pd*)

**lemma** *path-nret-ex-nipd*: **assumes**  $\langle is-path \ \pi \rangle$   $\langle \forall i. \ \pi \ i \neq return \rangle$  **shows**  $\langle \forall i. \ (\exists j \geq i. \ (\forall k > j. \ \pi \ k \neq ipd \ (\pi \ j))) \rangle$  **proof** (*rule, rule ccontr*)  
**fix**  $i$   
**assume**  $\langle \neg (\exists j \geq i. \ \forall k > j. \ \pi \ k \neq ipd \ (\pi \ j)) \rangle$   
**hence**  $*$ :  $\langle \forall j \geq i. \ (\exists k > j. \ \pi \ k = ipd \ (\pi \ j)) \rangle$  **by** *blast*  
**have**  $\langle \forall j. \ (\exists k > j. \ (\pi \ll i) \ k = ipd \ ((\pi \ll i) \ j)) \rangle$  **proof**  
**fix**  $j$   
**have**  $\langle i + j \geq i \rangle$  **by** *auto*  
**then obtain**  $k$  **where**  $k$ :  $\langle k > i + j \rangle$   $\langle \pi \ k = ipd \ (\pi \ (i + j)) \rangle$  **using**  $*$  **by** *blast*  
**hence**  $\langle (\pi \ll i) \ (k - i) = ipd \ ((\pi \ll i) \ j) \rangle$  **by** *auto*  
**moreover**  
**have**  $\langle k - i > j \rangle$  **using**  $k$  **by** *auto*  
**ultimately**  
**show**  $\langle \exists k > j. \ (\pi \ll i) \ k = ipd \ ((\pi \ll i) \ j) \rangle$  **by** *auto*  
**qed**  
**moreover**  
**have**  $\langle is-path \ (\pi \ll i) \rangle$  **using** *assms(1) path-path-shift* **by** *simp*  
**ultimately**  
**obtain**  $k$  **where**  $\langle (\pi \ll i) \ k = return \rangle$  **using** *all-ipd-imp-ret* **by** *blast*  
**thus**  $\langle False \rangle$  **using** *assms(2)* **by** *auto*  
**qed**

**lemma** *path-nret-ex-all-cd*: **assumes**  $\langle is-path \ \pi \rangle$   $\langle \forall i. \ \pi \ i \neq return \rangle$  **shows**  $\langle \forall i. \ (\exists j \geq i. \ (\forall k > j. \ k \ cd^\pi \rightarrow j)) \rangle$   
**unfolding** *is-cdi-def* **using** *assms path-nret-ex-nipd[OF assms]* **by** (*metis atLeastAtMost-iff ipd-not-self linorder-neqE-nat not-le path-nodes*)

**lemma** *path-nret-inf-all-cd*: **assumes**  $\langle is-path \ \pi \rangle$   $\langle \forall i. \ \pi \ i \neq return \rangle$  **shows**  $\langle \neg finite \ \{j. \ \forall k > j. \ k \ cd^\pi \rightarrow j\} \rangle$   
**using** *unbounded-nat-set-infinite path-nret-ex-all-cd[OF assms]* **by** *auto*

**lemma** *path-nret-inf-icd-seq*: **assumes** *path*:  $\langle is-path \ \pi \rangle$  **and** *nret*:  $\langle \forall i. \ \pi \ i \neq return \rangle$   
**obtains**  $f$  **where**  $\langle \forall i. \ f \ (Suc \ i) \ icd^\pi \rightarrow f \ i \rangle$   $\langle range \ f = \{i. \ \forall j > i. \ j \ cd^\pi \rightarrow i\} \rangle$   $\langle \neg (\exists i. \ f \ 0 \ cd^\pi \rightarrow i) \rangle$   
**proof** –

```

note path-nret-inf-all-cd[OF assms]
then obtain f where ran:  $\langle \text{range } f = \{j. \forall k > j. k \text{ cd}^\pi \rightarrow j\} \rangle$  and asc:  $\langle \forall i. f \ i < f \ (Suc \ i) \rangle$  using
infinite-ascending by blast
have mono:  $\langle \forall i \ j. i < j \longrightarrow f \ i < f \ j \rangle$  using asc by (metis lift-Suc-mono-less)
{
  fix i
  have cd:  $\langle f \ (Suc \ i) \text{ cd}^\pi \rightarrow f \ i \rangle$  using ran asc by auto
  have  $\langle f \ (Suc \ i) \text{ icd}^\pi \rightarrow f \ i \rangle$  proof (rule ccontr)
    assume  $\langle \neg f \ (Suc \ i) \text{ icd}^\pi \rightarrow f \ i \rangle$ 
    then obtain m where im:  $\langle f \ i < m \rangle$  and mi:  $\langle m < f \ (Suc \ i) \rangle$  and cdm:  $\langle f \ (Suc \ i) \text{ cd}^\pi \rightarrow m \rangle$  unfolding
is-cdi-def using assms(1) cd by auto
    have  $\langle \forall k > m. k \text{ cd}^\pi \rightarrow m \rangle$  proof (rule,rule,cases)
      fix k assume  $\langle f \ (Suc \ i) < k \rangle$ 
      hence  $\langle k \text{ cd}^\pi \rightarrow f \ (Suc \ i) \rangle$  using ran by auto
      thus  $\langle k \text{ cd}^\pi \rightarrow m \rangle$  using cdm cd-trans by metis
    next
      fix k assume mk:  $\langle m < k \rangle$  and  $\langle \neg f \ (Suc \ i) < k \rangle$ 
      hence ik:  $\langle k \leq f \ (Suc \ i) \rangle$  by simp
      thus  $\langle k \text{ cd}^\pi \rightarrow m \rangle$  using cdm by (metis cdi-prefix mk)
    qed
    hence  $\langle m \in \text{range } f \rangle$  using ran by blast
    then obtain j where m:  $\langle m = f \ j \rangle$  by blast
    show  $\langle False \rangle$  using im mi mono unfolding m by (metis Suc-lessI le-less not-le)
  qed
}
moreover
{
  fix m
  assume cdm:  $\langle f \ 0 \text{ cd}^\pi \rightarrow m \rangle$ 
  have  $\langle \forall k > m. k \text{ cd}^\pi \rightarrow m \rangle$  proof (rule,rule,cases)
    fix k assume  $\langle f \ 0 < k \rangle$ 
    hence  $\langle k \text{ cd}^\pi \rightarrow f \ 0 \rangle$  using ran by auto
    thus  $\langle k \text{ cd}^\pi \rightarrow m \rangle$  using cdm cd-trans by metis
  next
    fix k assume mk:  $\langle m < k \rangle$  and  $\langle \neg f \ 0 < k \rangle$ 
    hence ik:  $\langle k \leq f \ 0 \rangle$  by simp
    thus  $\langle k \text{ cd}^\pi \rightarrow m \rangle$  using cdm by (metis cdi-prefix mk)
  qed
  hence  $\langle m \in \text{range } f \rangle$  using ran by blast
  then obtain j where m:  $\langle m = f \ j \rangle$  by blast
  hence fj0:  $\langle f \ j < f \ 0 \rangle$  using cdm m is-cdi-def by auto
  hence  $\langle 0 < j \rangle$  by (metis less-irrefl neq0-conv)
  hence  $\langle False \rangle$  using fj0 mono by fastforce
}
ultimately show  $\langle thesis \rangle$  using that ran by blast
qed

```

**lemma** *cdi-iff-no-strict-pd*:  $\langle i \text{ cd}^\pi \rightarrow k \iff \text{is-path } \pi \wedge k < i \wedge \pi \ i \neq \text{return} \wedge (\forall j \in \{k..i\}. \neg (\pi \ k, \pi \ j) \in \text{pdt}) \rangle$

**proof**

```

assume cd:  $\langle i \text{ cd}^\pi \rightarrow k \rangle$ 
have 1:  $\langle \text{is-path } \pi \wedge k < i \wedge \pi \ i \neq \text{return} \rangle$  using cd unfolding is-cdi-def by auto
have 2:  $\langle \forall j \in \{k..i\}. \neg (\pi \ k, \pi \ j) \in \text{pdt} \rangle$  proof (rule ccontr)
  assume  $\langle \neg (\forall j \in \{k..i\}. (\pi \ k, \pi \ j) \notin \text{pdt}) \rangle$ 
  then obtain j where  $\langle j \in \{k..i\} \rangle$  and  $\langle (\pi \ k, \pi \ j) \in \text{pdt} \rangle$  by auto
  hence  $\langle \pi \ j \neq \pi \ k \rangle$  and  $\langle \pi \ j \text{ pd} \rightarrow \pi \ k \rangle$  unfolding pdt-def by auto

```

**thus**  $\langle False \rangle$  **using** *path-pd-ipd* **by** (*metis*  $\langle j \in \{k..i\} \rangle$  *atLeastAtMost-iff cd cd-not-pd cdi-prefix le-eq-less-or-eq*)

**qed**

**show**  $\langle is-path \pi \wedge k < i \wedge \pi i \neq return \wedge (\forall j \in \{k..i\}. \neg (\pi k, \pi j) \in pdt) \rangle$  **using** *1 2* **by** *simp*

**next**

**assume**  $\langle is-path \pi \wedge k < i \wedge \pi i \neq return \wedge (\forall j \in \{k..i\}. \neg (\pi k, \pi j) \in pdt) \rangle$

**thus**  $\langle i cd^\pi \rightarrow k \rangle$  **by** (*metis ipd-in-pdt term-path-stable less-or-eq-imp-le not-cd-impl-ipd path-nodes*)

**qed**

## 2.5 Facts about Control Slices

**lemma** *last-cs*:  $\langle last (cs^\pi i) = \pi i \rangle$  **by** *auto*

**lemma** *cs-not-nil*:  $\langle cs^\pi n \neq [] \rangle$  **by** (*auto*)

**lemma** *cs-return*: **assumes**  $\langle \pi n = return \rangle$  **shows**  $\langle cs^\pi n = [\pi n] \rangle$  **by** (*metis assms cs.elims icd-imp-cd ret-no-cd*)

**lemma** *cs-0[simp]*:  $\langle cs^\pi 0 = [\pi 0] \rangle$  **using** *is-icdi-def is-cdi-def* **by** *auto*

**lemma** *cs-inj*: **assumes**  $\langle is-path \pi \rangle$   $\langle \pi n \neq return \rangle$   $\langle cs^\pi n = cs^\pi n' \rangle$  **shows**  $\langle n = n' \rangle$

**using** *assms* **proof** (*induction*  $\langle cs^\pi n \rangle$  *arbitrary*:  $\langle \pi \rangle$   $\langle n \rangle$   $\langle n' \rangle$  *rule:rev-induct*)

**case** *Nil* **hence**  $\langle False \rangle$  **using** *cs-not-nil* **by** *metis* **thus**  $\langle ?case \rangle$  **by** *simp*

**next**

**case** (*snoc* *x xs*  $\pi n n'$ ) **show**  $\langle ?case \rangle$  **proof** (*cases*  $\langle xs \rangle$ )

**case** *Nil*

**hence**  $*$ :  $\langle \neg (\exists k. n icd^\pi \rightarrow k) \rangle$  **using** *snoc(2)* *cs-not-nil*

**by** (*auto,metis append1-eq-conv append-Nil cs-not-nil*)

**moreover**

**have**  $\langle [x] = cs^\pi n' \rangle$  **using** *Nil snoc* **by** *auto*

**hence**  $**$ :  $\langle \neg (\exists k. n' icd^\pi \rightarrow k) \rangle$  **using** *cs-not-nil*

**by** (*auto,metis append1-eq-conv append-Nil cs-not-nil*)

**ultimately**

**have**  $\langle \forall k. \neg n cd^\pi \rightarrow k \rangle$   $\langle \forall k. \neg n' cd^\pi \rightarrow k \rangle$  **using** *excd-impl-excd* **by** *auto blast+*

**moreover**

**hence**  $\langle \pi n = \pi n' \rangle$  **using** *snoc(5,2)* **by** *auto* (*metis \* \*\* list.inject*)

**ultimately**

**show**  $\langle n = n' \rangle$  **using** *other-claim' snoc* **by** *blast*

**next**

**case** (*Cons* *y ys*)

**hence**  $*$ :  $\langle \exists k. n icd^\pi \rightarrow k \rangle$  **using** *snoc(2)* **by** *auto* (*metis append-is-Nil-conv list.distinct(1) list.inject*)

**then obtain** *k* **where**  $k$ :  $\langle n icd^\pi \rightarrow k \rangle$  **by** *auto*

**have**  $\langle k = (THE k . n icd^\pi \rightarrow k) \rangle$  **using** *k* **by** (*metis icd-is-the-icd*)

**hence** *xsk*:  $\langle xs = cs^\pi k \rangle$  **using**  $*$  *k snoc(2)* **unfolding** *cs.simps[of*  $\langle \pi \rangle$   $\langle n \rangle$  *]* **by** *auto*

**have**  $**$ :  $\langle \exists k. n' icd^\pi \rightarrow k \rangle$  **using** *snoc(2)[unfolded snoc(5)]* **by** *auto* (*metis Cons append1-eq-conv append-Nil list.distinct(1)*)

**then obtain** *k'* **where**  $k'$ :  $\langle n' icd^\pi \rightarrow k' \rangle$  **by** *auto*

**hence**  $\langle k' = (THE k' . n' icd^\pi \rightarrow k') \rangle$  **using**  $k'$  **by** (*metis icd-is-the-icd*)

**hence** *xsk'*:  $\langle xs = cs^\pi k' \rangle$  **using**  $**$  *k' snoc(5,2)* **unfolding** *cs.simps[of*  $\langle \pi \rangle$   $\langle n' \rangle$  *]* **by** *auto*

**hence**  $\langle cs^\pi k = cs^\pi k' \rangle$  **using** *xsk* **by** *simp*

**moreover**

**have** *kn*:  $\langle k < n \rangle$  **using** *k* **by** (*metis is-icdi-def is-cdi-def*)

**hence**  $\langle \pi k \neq return \rangle$  **using** *snoc* **by** (*metis term-path-stable less-imp-le*)

**ultimately**

**have** *kk'[simp]*:  $\langle k' = k \rangle$  **using** *snoc(1)* *xsk snoc(3)* **by** *metis*

**have** *nk0*:  $\langle n - k icd^\pi \rightarrow 0 \rangle$   $\langle n' - k' icd^\pi \rightarrow 0 \rangle$  **using** *k k' icd-path-shift0 snoc(3)* **by** *auto*

**moreover**  
**have**  $\langle nkr: \langle \pi \ll k \rangle (n-k) \neq \text{return} \rangle$  **using** *snoc(4)* **kn** **by** *auto*  
**moreover**  
**have**  $\langle \text{is-path } (\pi \ll k) \rangle$  **by** (*metis path-path-shift snoc.prem(1)*)  
**moreover**  
**have**  $\langle kn': \langle k < n' \rangle$  **using**  $k' kk'$  **by** (*metis is-icdi-def is-cdi-def*)  
**have**  $\langle \pi n = \pi n' \rangle$  **using** *snoc(5)* **\*\*** **by** *auto*  
**hence**  $\langle (\pi \ll k)(n-k) = (\pi \ll k)(n'-k) \rangle$  **using** *kn kn'* **by** *auto*  
**ultimately**  
**have**  $\langle n - k = n' - k \rangle$  **using** *other-claim* **by** *auto*  
**thus**  $\langle n = n' \rangle$  **using** *kn kn'* **by** *auto*  
**qed**  
**qed**

**lemma** *cs-cases*: **fixes**  $\pi i$   
**obtains** (*base*)  $\langle cs^\pi i = [\pi i] \rangle$  **and**  $\langle \forall k. \neg i \text{ cd}^\pi \rightarrow k \rangle$  |  
(*depend*)  $k$  **where**  $\langle cs^\pi i = (cs^\pi k) @ [\pi i] \rangle$  **and**  $\langle i \text{ icd}^\pi \rightarrow k \rangle$   
**proof** *cases*  
**assume** \*:  $\langle \exists k. i \text{ icd}^\pi \rightarrow k \rangle$   
**then obtain**  $k$  **where**  $k: \langle i \text{ icd}^\pi \rightarrow k \rangle$  ..  
**hence**  $\langle k = (\text{THE } k. i \text{ icd}^\pi \rightarrow k) \rangle$  **by** (*metis icd-is-the-icd*)  
**hence**  $\langle cs^\pi i = (cs^\pi k) @ [\pi i] \rangle$  **using** \* **by** *auto*  
**with**  $k$  **that show**  $\langle \text{thesis} \rangle$  **by** *simp*

**next**  
**assume** \*:  $\langle \neg (\exists k. i \text{ icd}^\pi \rightarrow k) \rangle$   
**hence**  $\langle \forall k. \neg i \text{ cd}^\pi \rightarrow k \rangle$  **by** (*metis excd-impl-exicd*)  
**moreover**  
**have**  $\langle cs^\pi i = [\pi i] \rangle$  **using** \* **by** *auto*  
**ultimately**  
**show**  $\langle \text{thesis} \rangle$  **using** *that* **by** *simp*  
**qed**

**lemma** *cs-length-one*: **assumes**  $\langle \text{length } (cs^\pi i) = 1 \rangle$  **shows**  $\langle cs^\pi i = [\pi i] \rangle$  **and**  $\langle \forall k. \neg i \text{ cd}^\pi \rightarrow k \rangle$   
**apply** (*cases i*)  $\langle \pi \rangle$  **rule:** *cs-cases*  
**using** *assms cs-not-nil*  
**apply** *auto*  
**apply** (*cases i*)  $\langle \pi \rangle$  **rule:** *cs-cases*  
**using** *assms cs-not-nil*  
**by** *auto*

**lemma** *cs-length-g-one*: **assumes**  $\langle \text{length } (cs^\pi i) \neq 1 \rangle$  **obtains**  $k$  **where**  $\langle cs^\pi i = (cs^\pi k) @ [\pi i] \rangle$  **and**  $\langle i \text{ icd}^\pi \rightarrow k \rangle$   
**apply** (*cases i*)  $\langle \pi \rangle$  **rule:** *cs-cases*  
**using** *assms cs-not-nil* **by** *auto*

**lemma** *claim*: **assumes** *path*:  $\langle \text{is-path } \pi \rangle$   $\langle \text{is-path } \pi' \rangle$  **and** *ii*:  $\langle cs^\pi i = cs^{\pi'} i' \rangle$  **and** *jj*:  $\langle cs^\pi j = cs^{\pi'} j' \rangle$   
**and** *bl*:  $\langle \text{butlast } (cs^\pi i) = \text{butlast } (cs^\pi j) \rangle$  **and** *nret*:  $\langle \pi i \neq \text{return} \rangle$  **and** *ilj*:  $\langle i < j \rangle$   
**shows**  $\langle i' < j' \rangle$   
**proof** (*cases*)  
**assume** \*:  $\langle \text{length } (cs^\pi i) = 1 \rangle$   
**hence** \*\*:  $\langle \text{length } (cs^\pi i) = 1 \rangle$   $\langle \text{length } (cs^\pi j) = 1 \rangle$   $\langle \text{length } (cs^{\pi'} i') = 1 \rangle$   $\langle \text{length } (cs^{\pi'} j') = 1 \rangle$   
**apply** *metis*  
**apply** (*metis \* bl butlast.simps(2) butlast-snoc cs-length-g-one cs-length-one(1) cs-not-nil*)  
**apply** (*metis \* ii*)  
**by** (*metis \* bl butlast.simps(2) butlast-snoc cs-length-g-one cs-length-one(1) cs-not-nil jj*)

**then obtain**  $\langle cs^\pi i = [\pi i] \rangle \langle cs^\pi j = [\pi j] \rangle \langle cs^{\pi'} j' = [\pi' j'] \rangle \langle cs^{\pi'} i' = [\pi' i'] \rangle$   
 $\langle \forall k. \neg j \text{ cd}^\pi \rightarrow k \rangle \langle \forall k. \neg i' \text{ cd}^{\pi'} \rightarrow k \rangle \langle \forall k. \neg j' \text{ cd}^{\pi'} \rightarrow k \rangle$   
**by** (*metis cs-length-one \*\**)  
**moreover**  
**hence**  $\langle \pi i = \pi' i' \rangle \langle \pi j = \pi' j' \rangle$  **using** *assms by auto*  
**ultimately**  
**show**  $\langle i' < j' \rangle$  **using** *nret ilj path claim'' by blast*  
**next**  
**assume** \*:  $\langle \text{length}(cs^\pi i) \neq 1 \rangle$   
**hence** \*\*:  $\langle \text{length}(cs^\pi i) \neq 1 \rangle \langle \text{length}(cs^\pi j) \neq 1 \rangle \langle \text{length}(cs^{\pi'} i') \neq 1 \rangle \langle \text{length}(cs^{\pi'} j') \neq 1 \rangle$   
**apply** *metis*  
**apply** (*metis \* bl butlast.simps(2) butlast-snoc cs-length-g-one cs-length-one(1) cs-not-nil*)  
**apply** (*metis \* ii*)  
**by** (*metis \* bl butlast.simps(2) butlast-snoc cs-length-g-one cs-length-one(1) cs-not-nil jj*)  
**obtain**  $k \ l \ k' \ l'$  **where** \*\*\*:  
 $\langle cs^\pi i = (cs^\pi k) @ [\pi i] \rangle \langle cs^\pi j = (cs^\pi l) @ [\pi j] \rangle \langle cs^{\pi'} i' = (cs^{\pi'} k') @ [\pi' i'] \rangle \langle cs^{\pi'} j' = (cs^{\pi'} l') @ [\pi' j'] \rangle$   
**and**  
 $\langle i \text{ icd}^\pi \rightarrow k \rangle \langle j \text{ icd}^\pi \rightarrow l \rangle \langle i' \text{ icd}^{\pi'} \rightarrow k' \rangle \langle j' \text{ icd}^{\pi'} \rightarrow l' \rangle$   
**by** (*metis \*\* cs-length-g-one*)  
**hence**  $\langle cs^\pi k = cs^\pi l \rangle \langle cs^{\pi'} k' = cs^{\pi'} l' \rangle$  **using** *assms by auto*  
**moreover**  
**have**  $\langle \pi k \neq \text{return} \rangle \langle \pi' k' \neq \text{return} \rangle$  **using** *nret*  
**apply** (*metis is-icdi-def icds(1) is-cdi-def term-path-stable less-imp-le*)  
**by** (*metis is-cdi-def is-icdi-def icds(3) term-path-stable less-imp-le*)  
**ultimately**  
**have**  $lk[simp]: \langle l = k \rangle \langle l' = k' \rangle$  **using** *path cs-inj by auto*  
**let**  $\langle ?\pi \rangle = \langle \pi \ll k \rangle$   
**let**  $\langle ?\pi' \rangle = \langle \pi' \ll k' \rangle$   
**have**  $\langle i-k \text{ icd}^{?\pi} \rightarrow 0 \rangle \langle j-k \text{ icd}^{?\pi} \rightarrow 0 \rangle \langle i'-k' \text{ icd}^{?\pi'} \rightarrow 0 \rangle \langle j'-k' \text{ icd}^{?\pi'} \rightarrow 0 \rangle$  **using** *icd-path-shift0 path icds*  
**by auto**  
**moreover**  
**have**  $ki: \langle k < i \rangle$  **using** *icds by (metis is-icdi-def is-cdi-def)*  
**hence**  $\langle i-k < j-k \rangle$  **by** (*metis diff-is-0-eq diff-less-mono ilj nat-le-linear order.strict-trans*)  
**moreover**  
**have**  $\pi i: \langle \pi i = \pi' i' \rangle \langle \pi j = \pi' j' \rangle$  **using** *assms \*\*\* by auto*  
**have**  $\langle k' < i' \rangle \langle k' < j' \rangle$  **using** *icds unfolding lk by (metis is-cdi-def is-icdi-def)+*  
**hence**  $\langle ?\pi (i-k) = ?\pi' (i'-k') \rangle \langle ?\pi (j-k) = ?\pi' (j'-k') \rangle$  **using**  $\pi i \ ki \ ilj$  **by auto**  
**moreover**  
**have**  $\langle ?\pi (i-k) \neq \text{return} \rangle$  **using** *nret ki by auto*  
**moreover**  
**have**  $\langle \text{is-path } ?\pi \rangle \langle \text{is-path } ?\pi' \rangle$  **using** *path path-path-shift by auto*  
**ultimately**  
**have**  $\langle i'-k' < j' - k' \rangle$  **using** *claim' by blast*  
**thus**  $\langle i' < j' \rangle$  **by** (*metis diff-is-0-eq diff-less-mono less-nat-zero-code linorder-neqE-nat nat-le-linear*)  
**qed**

**lemma** *cs-split'*: **assumes**  $\langle cs^\pi i = xs @ [x, x'] @ ys \rangle$  **shows**  $\langle \exists m. cs^\pi m = xs @ [x] \wedge i \text{ cd}^\pi \rightarrow m \rangle$   
**using** *assms proof (induction <ys> arbitrary: <i> rule:rev-induct)*  
**case** (*snoc y ys*)  
**hence**  $\langle \text{length}(cs^\pi i) \neq 1 \rangle$  **by auto**  
**then obtain**  $i'$  **where**  $\langle cs^\pi i = (cs^\pi i') @ [\pi i] \rangle$  **and** \*:  $\langle i \text{ icd}^\pi \rightarrow i' \rangle$  **using** *cs-length-g-one[of <\pi> <i>] by metis*  
**hence**  $\langle cs^\pi i' = xs @ [x, x'] @ ys \rangle$  **using** *snoc(2) by (metis append1-eq-conv append-assoc)*  
**then obtain**  $m$  **where** \*\*:  $\langle cs^\pi m = xs @ [x] \rangle$  **and**  $\langle i' \text{ cd}^\pi \rightarrow m \rangle$  **using** *snoc(1) by blast*  
**hence**  $\langle i \text{ cd}^\pi \rightarrow m \rangle$  **using** \* *cd-trans by (metis is-icdi-def)*



**with** **\*\* show**  $\langle ?case \rangle$  **by** *blast*  
**next**  
**case** *Nil*  
**hence**  $\langle length (cs^\pi i) \neq 1 \rangle$  **by** *auto*  
**then obtain**  $i'$  **where**  $a: \langle cs^\pi i = (cs^\pi i') @ [\pi i] \rangle$  **and**  $*$ :  $\langle i \text{ icd}^\pi \rightarrow i' \rangle$  **using** *cs-length-g-one*[*of*  $\langle \pi \rangle \langle i \rangle$ ] **by** *metis*  
**have**  $\langle cs^\pi i = (xs@[x])@[x'] \rangle$  **using** *Nil* **by** *auto*  
**hence**  $\langle cs^\pi i' = xs@[x] \rangle$  **using** *append1-eq-conv*  $a$  **by** *metis*  
**thus**  $\langle ?case \rangle$  **using**  $*$  **unfolding** *is-icdi-def* **by** *blast*  
**qed**

**lemma** *cs-split*: **assumes**  $\langle cs^\pi i = xs@[x]@ys@[ \pi i] \rangle$  **shows**  $\langle \exists m. cs^\pi m = xs@[x] \wedge i \text{ cd}^\pi \rightarrow m \rangle$  **proof** –  
**obtain**  $x' \text{ ys}'$  **where**  $\langle ys@[ \pi i] = [x']@ys' \rangle$  **by** (*metis append-Cons append-Nil neg-Nil-conv*)  
**thus**  $\langle ?thesis \rangle$  **using** *cs-split'*[*of*  $\langle \pi \rangle \langle i \rangle \langle xs \rangle \langle x \rangle \langle x' \rangle \langle ys' \rangle$ ] **assms** **by** *auto*  
**qed**

**lemma** *cs-less-split*: **assumes**  $\langle xs \prec ys \rangle$  **obtains**  $a$  **as** **where**  $\langle ys = xs@a\#as \rangle$   
**using** *assms* **unfolding** *cs-less.simps* **apply** *auto*  
**by** (*metis Cons-nth-drop-Suc append-take-drop-id*)

**lemma** *cs-select-is-cs*: **assumes**  $\langle is\text{-path } \pi \rangle \langle xs \neq Nil \rangle \langle xs \prec cs^\pi k \rangle$  **shows**  $\langle cs^\pi (\pi \downarrow xs) = xs \rangle \langle k \text{ cd}^\pi \rightarrow (\pi \downarrow xs) \rangle$  **proof** –  
**obtain**  $b \text{ bs}$  **where**  $b: \langle cs^\pi k = xs@a\#bs \rangle$  **using** *assms* *cs-less-split* **by** *blast*  
**obtain**  $a$  **as** **where**  $a: \langle xs = as@[a] \rangle$  **using** *assms* **by** (*metis rev-exhaust*)  
**have**  $\langle cs^\pi k = as@[a,b]@bs \rangle$  **using**  $a \text{ b}$  **by** *auto*  
**then obtain**  $k'$  **where**  $csk: \langle cs^\pi k' = xs \rangle$  **and** *is-cd*:  $\langle k \text{ cd}^\pi \rightarrow k' \rangle$  **using** *cs-split'*  $a$  **by** *blast*  
**hence** *nret*:  $\langle \pi k' \neq return \rangle$  **by** (*metis is-cdi-def term-path-stable less-imp-le*)  
**show**  $a: \langle cs^\pi (\pi \downarrow xs) = xs \rangle$  **unfolding** *cs-select-def* **using** *cs-inj*[*OF* *assms*(1) *nret*] *csk* *the-equality*[*of* -  $\langle k' \rangle$ ]  
**by** (*metis (mono-tags)*)  
**show**  $\langle k \text{ cd}^\pi \rightarrow (\pi \downarrow xs) \rangle$  **unfolding** *cs-select-def* **by** (*metis a* *assms*(1) *cs-inj* *cs-select-def* *csk* *is-cd* *nret*)  
**qed**

**lemma** *cd-in-cs*: **assumes**  $\langle n \text{ cd}^\pi \rightarrow m \rangle$  **shows**  $\langle \exists ns. cs^\pi n = (cs^\pi m) @ ns @ [\pi n] \rangle$   
**using** *assms* **proof** (*induction rule: cd-induct*)  
**case** (*base*  $n$ ) **thus**  $\langle ?case \rangle$  **by** (*metis append-Nil cs.simps icd-is-the-icd*)  
**next**  
**case** (*IS*  $k \text{ n}$ )  
**hence**  $\langle cs^\pi n = cs^\pi k @ [\pi n] \rangle$  **by** (*metis cs.simps icd-is-the-icd*)  
**thus**  $\langle ?case \rangle$  **using** *IS* **by** *force*  
**qed**

**lemma** *butlast-cs-not-cd*: **assumes**  $\langle butlast (cs^\pi m) = butlast (cs^\pi n) \rangle$  **shows**  $\langle \neg m \text{ cd}^\pi \rightarrow n \rangle$   
**by** (*metis append-Cons append-Nil append-assoc assms* *cd-in-cs* *cs-not-nil* *list.distinct*(1) *self-append-conv* *snoc-eq-iff-butlast*)

**lemma** *wn-cs-butlast*: **assumes**  $\langle butlast (cs^\pi m) = butlast (cs^\pi n) \rangle \langle i \text{ cd}^\pi \rightarrow m \rangle \langle j \text{ cd}^\pi \rightarrow n \rangle \langle m < n \rangle$  **shows**  $\langle i < j \rangle$   
**proof** (*rule ccontr*)  
**assume**  $\langle \neg i < j \rangle$   
**moreover**  
**have**  $\langle \neg n \text{ cd}^\pi \rightarrow m \rangle$  **by** (*metis assms*(1) *butlast-cs-not-cd*)  
**ultimately**  
**have**  $\langle n \leq m \rangle$  **using** *assms*(2,3) *cr-wn''* **by** *auto*  
**thus**  $\langle False \rangle$  **using** *assms*(4) **by** *auto*  
**qed**

This is the central theorem making the control slice suitable for matching indices between executions.

**theorem** *cs-order*: **assumes** *path*:  $\langle is\text{-path } \pi \rangle \langle is\text{-path } \pi' \rangle$  **and** *csi*:  $\langle cs^\pi i = cs^{\pi'} i \rangle$   
**and** *csj*:  $\langle cs^\pi j = cs^{\pi'} j \rangle$  **and** *nret*:  $\langle \pi i \neq \text{return} \rangle$  **and** *ilj*:  $\langle i < j \rangle$   
**shows**  $\langle i' < j' \rangle$

**proof** –

**have**  $\langle cs^\pi i \neq cs^\pi j \rangle$  **using** *cs-inj*[*OF path*(1) *nret*] *ilj* **by** *blast*  
**moreover**

**have**  $\langle cs^\pi i \neq Nil \rangle \langle cs^\pi j \neq Nil \rangle$  **by** (*metis cs-not-nil*)+

**ultimately show**  $\langle ?thesis \rangle$  **proof** (*cases rule: list-neq-prefix-cases*)

**case** (*diverge xs x x' ys ys'*)

**note** *csx* =  $\langle cs^\pi i = xs @ [x] @ ys \rangle$

**note** *csx'* =  $\langle cs^\pi j = xs @ [x'] @ ys' \rangle$

**note** *xx* =  $\langle x \neq x' \rangle$

**show**  $\langle i' < j' \rangle$  **proof** (*cases ys*)

**assume** *ys*:  $\langle ys = Nil \rangle$

**show**  $\langle ?thesis \rangle$  **proof** (*cases ys'*)

**assume** *ys'*:  $\langle ys' = Nil \rangle$

**have** *cs*:  $\langle cs^\pi i = xs @ [x] \rangle \langle cs^\pi j = xs @ [x'] \rangle$  **by** (*metis append-Nil2 csx ys, metis append-Nil2 csx' ys'*)

**hence** *bl*:  $\langle butlast (cs^\pi i) = butlast (cs^\pi j) \rangle$  **by** *auto*

**show**  $\langle i' < j' \rangle$  **using** *claim*[*OF path csi csj bl nret ilj*] .

**next**

**fix** *y' zs'*

**assume** *ys'*:  $\langle ys' = y' \# zs' \rangle$

**have** *cs*:  $\langle cs^\pi i = xs @ [x] \rangle \langle cs^\pi j = xs @ [x', y'] @ zs' \rangle$  **by** (*metis append-Nil2 csx ys, metis append-Cons append-Nil csx' ys'*)

**obtain** *n* **where** *n*:  $\langle cs^\pi n = xs @ [x'] \rangle$  **and** *jn*:  $\langle j \text{ cd}^\pi \rightarrow n \rangle$  **using** *cs cs-split'* **by** *blast*

**obtain** *n'* **where** *n'*:  $\langle cs^{\pi'} n' = xs @ [x'] \rangle$  **and** *jn'*:  $\langle j' \text{ cd}^{\pi'} \rightarrow n' \rangle$  **using** *cs cs-split' unfolding csj* **by**

*blast*

**have** *csn* :  $\langle cs^\pi n = cs^{\pi'} n' \rangle$  **and** *bl*:  $\langle butlast (cs^\pi i) = butlast (cs^\pi n) \rangle$  **using** *n n' cs* **by** *auto*

**hence** *bl'*:  $\langle butlast (cs^{\pi'} i') = butlast (cs^{\pi'} n') \rangle$  **using** *csi* **by** *auto*

**have** *notcd*:  $\langle \neg i \text{ cd}^\pi \rightarrow n \rangle$  **by** (*metis butlast-cs-not-cd bl*)

**have** *nin*:  $\langle i \neq n \rangle$  **using** *cs n xx* **by** *auto*

**have** *iln*:  $\langle i < n \rangle$  **apply** (*rule ccontr*) **using** *cr-wn'*[*OF jn notcd*] *nin ilj* **by** *auto*

**note** *claim*[*OF path csi csn bl nret iln*]

**hence**  $\langle i' < n' \rangle$  .

**thus**  $\langle i' < j' \rangle$  **using** *jn' unfolding is-cdi-def* **by** *auto*

**qed**

**next**

**fix** *y zs*

**assume** *ys*:  $\langle ys = y \# zs \rangle$

**show**  $\langle ?thesis \rangle$  **proof** (*cases ys'*)

**assume** *ys'* :  $\langle ys' = Nil \rangle$

**have** *cs*:  $\langle cs^\pi i = xs @ [x, y] @ zs \rangle \langle cs^\pi j = xs @ [x'] \rangle$  **by** (*metis append-Cons append-Nil csx ys, metis append-Nil2 csx' ys'*)

**obtain** *n* **where** *n*:  $\langle cs^\pi n = xs @ [x] \rangle$  **and** *jn*:  $\langle i \text{ cd}^\pi \rightarrow n \rangle$  **using** *cs cs-split'* **by** *blast*

**obtain** *n'* **where** *n'*:  $\langle cs^{\pi'} n' = xs @ [x] \rangle$  **and** *jn'*:  $\langle i' \text{ cd}^{\pi'} \rightarrow n' \rangle$  **using** *cs cs-split' unfolding csi* **by** *blast*

**have** *csn* :  $\langle cs^\pi n = cs^{\pi'} n' \rangle$  **and** *bl*:  $\langle butlast (cs^\pi n) = butlast (cs^{\pi'} j) \rangle$  **using** *n n' cs* **by** *auto*

**hence** *bl'*:  $\langle butlast (cs^{\pi'} j') = butlast (cs^{\pi'} n') \rangle$  **using** *csj* **by** *auto*

**have** *notcd*:  $\langle \neg j' \text{ cd}^{\pi'} \rightarrow n' \rangle$  **by** (*metis butlast-cs-not-cd bl'*)

**have** *nin*:  $\langle n < i \rangle$  **using** *jn unfolding is-cdi-def* **by** *auto*

**have** *nlj*:  $\langle n < j \rangle$  **using** *nin ilj* **by** *auto*

**note** *claim*[*OF path csn csj bl - nlj*]

**hence** *nj'*:  $\langle n' < j' \rangle$  **using** *term-path-stable*[*OF path*(1) -] *less-imp-le nin nret* **by** *auto*

**show**  $\langle i' < j' \rangle$  **apply**(*rule ccontr*) **using** *cdi-prefix*[*OF jn' nj'*] *notcd* **by** *auto*

**next**

**fix**  $y' zs'$   
**assume**  $ys' : \langle ys' = y' \# zs' \rangle$   
**have**  $cs : \langle cs^\pi i = xs@[x,y]@zs \rangle \langle cs^\pi j = xs@[x',y']@zs' \rangle$  **by** (*metis append-Cons append-Nil csx ys,metis append-Cons append-Nil csx' ys'*)  
**have**  $neq : \langle cs^\pi i \neq cs^\pi j \rangle$  **using** *cs-inj path nret ilj* **by** *blast*  
**obtain**  $m$  **where**  $m : \langle cs^\pi m = xs@[x] \rangle$  **and**  $im : \langle i \text{ cd}^\pi \rightarrow m \rangle$  **using** *cs cs-split'* **by** *blast*  
**obtain**  $n$  **where**  $n : \langle cs^\pi n = xs@[x'] \rangle$  **and**  $jn : \langle j \text{ cd}^\pi \rightarrow n \rangle$  **using** *cs cs-split'* **by** *blast*  
**obtain**  $m'$  **where**  $m' : \langle cs^{\pi'} m' = xs@[x] \rangle$  **and**  $im' : \langle i' \text{ cd}^{\pi'} \rightarrow m' \rangle$  **using** *cs cs-split'* **unfolding** *csi* **by** *blast*  
**obtain**  $n'$  **where**  $n' : \langle cs^{\pi'} n' = xs@[x'] \rangle$  **and**  $jn' : \langle j' \text{ cd}^{\pi'} \rightarrow n' \rangle$  **using** *cs cs-split'* **unfolding** *csj* **by** *blast*  
**have**  $\langle m \leq n \rangle$  **using** *ilj m n wn-cs-butlast[OF - jn im]* **by** *force*  
**moreover**  
**have**  $\langle m \neq n \rangle$  **using** *m n xx* **by** (*metis last-snoc*)  
**ultimately**  
**have**  $mn : \langle m < n \rangle$  **by** *auto*  
**moreover**  
**have**  $\langle \pi m \neq \text{return} \rangle$  **by** (*metis last-cs last-snoc m mn n path(1) term-path-stable xx less-imp-le*)  
**moreover**  
**have**  $\langle \text{butlast } (cs^\pi m) = \text{butlast } (cs^\pi n) \rangle \langle cs^\pi m = cs^{\pi'} m' \rangle \langle cs^\pi n = cs^{\pi'} n' \rangle$  **using** *m n n' m'* **by** *auto*  
**ultimately**  
**have**  $\langle m' < n' \rangle$  **using** *claim path* **by** *blast*  
**thus**  $\langle i' < j' \rangle$  **using** *m' n' im' jn' wn-cs-butlast* **by** (*metis butlast-snoc*)  
**qed**  
**qed**  
**next**  
**case** (*prefix1 xs*)  
**note**  $pfx = \langle cs^\pi i = cs^\pi j @ xs \rangle$   
**note**  $xs = \langle xs \neq [] \rangle$   
**obtain**  $a \text{ as}$  **where**  $\langle xs = a \# as \rangle$  **using** *xs* **by** (*metis list.exhaust*)  
**moreover**  
**obtain**  $bs \ b$  **where**  $bj : \langle cs^\pi j = bs@[b] \rangle$  **using** *cs-not-nil* **by** (*metis rev-exhaust*)  
**ultimately**  
**have**  $\langle cs^\pi i = bs@[b,a]@as \rangle$  **using** *pfx* **by** *auto*  
**then obtain**  $m$  **where**  $\langle cs^\pi m = bs@[b] \rangle$  **and**  $cdep : \langle i \text{ cd}^\pi \rightarrow m \rangle$  **using** *cs-split'* **by** *blast*  
**hence**  $mi : \langle m = j \rangle$  **using** *bj cs-inj* **by** (*metis is-cdi-def term-path-stable less-imp-le*)  
**hence**  $\langle i \text{ cd}^\pi \rightarrow j \rangle$  **using** *cdep* **by** *auto*  
**hence**  $\langle \text{False} \rangle$  **using** *ilj unfolding is-cdi-def* **by** *auto*  
**thus**  $\langle i' < j' \rangle \dots$   
**next**  
**case** (*prefix2 xs*)  
**have**  $pfx : \langle cs^{\pi'} i' @ xs = cs^{\pi'} j' \rangle$  **using** *prefix2 csi csj* **by** *auto*  
**note**  $xs = \langle xs \neq [] \rangle$   
**obtain**  $a \text{ as}$  **where**  $\langle xs = a \# as \rangle$  **using** *xs* **by** (*metis list.exhaust*)  
**moreover**  
**obtain**  $bs \ b$  **where**  $bj : \langle cs^{\pi'} i' = bs@[b] \rangle$  **using** *cs-not-nil* **by** (*metis rev-exhaust*)  
**ultimately**  
**have**  $\langle cs^{\pi'} j' = bs@[b,a]@as \rangle$  **using** *pfx* **by** *auto*  
**then obtain**  $m$  **where**  $\langle cs^{\pi'} m = bs@[b] \rangle$  **and**  $cdep : \langle j' \text{ cd}^{\pi'} \rightarrow m \rangle$  **using** *cs-split'* **by** *blast*  
**hence**  $mi : \langle m = i' \rangle$  **using** *bj cs-inj* **by** (*metis is-cdi-def term-path-stable less-imp-le*)  
**hence**  $\langle j' \text{ cd}^{\pi'} \rightarrow i' \rangle$  **using** *cdep* **by** *auto*  
**thus**  $\langle i' < j' \rangle$  **unfolding** *is-cdi-def* **by** *auto*  
**qed**  
**qed**

**lemma cs-order-le:** **assumes**  $\langle is\text{-path } \pi \rangle \langle is\text{-path } \pi' \rangle$  **and**  $csi: \langle cs^\pi i = cs^{\pi'} i' \rangle$   
**and**  $csj: \langle cs^\pi j = cs^{\pi'} j' \rangle$  **and**  $nret: \langle \pi i \neq return \rangle$  **and**  $ilj: \langle i \leq j \rangle$   
**shows**  $\langle i' \leq j' \rangle$  **proof cases**

**assume**  $\langle i < j \rangle$  **with**  $cs\text{-order}[OF\ assms(1,2,3,4,5)]$  **show**  $\langle ?thesis \rangle$  **by simp**  
**next**  
**assume**  $\langle \neg i < j \rangle$   
**hence**  $\langle i = j \rangle$  **using**  $ilj$  **by simp**  
**hence**  $csij: \langle cs^{\pi'} i' = cs^{\pi'} j' \rangle$  **using**  $csi\ csj$  **by simp**  
**have**  $nret': \langle \pi' i' \neq return \rangle$  **using**  $nret\ last\text{-cs}\ csi$  **by metis**  
**show**  $\langle ?thesis \rangle$  **using**  $cs\text{-inj}[OF\ path(2)\ nret'\ csij]$  **by simp**  
**qed**

**lemmas**  $cs\text{-induct}[case\text{-names } cs] = cs.\text{induct}$

**lemma icdi-path-swap:** **assumes**  $\langle is\text{-path } \pi' \rangle \langle j\ icd^{\pi'} \rightarrow k \rangle \langle \pi =_j \pi' \rangle$  **shows**  $\langle j\ icd^\pi \rightarrow k \rangle$  **using**  $assms\ unfolding\ eq\text{-up}\text{-to}\text{-def}\ is\text{-icdi}\text{-def}\ is\text{-cdi}\text{-def}$  **by auto**

**lemma icdi-path-swap-le:** **assumes**  $\langle is\text{-path } \pi' \rangle \langle j\ icd^{\pi'} \rightarrow k \rangle \langle \pi =_n \pi' \rangle \langle j \leq n \rangle$  **shows**  $\langle j\ icd^\pi \rightarrow k \rangle$  **by** ( $metis\ assms\ icdi\text{-path}\text{-swap}\ eq\text{-up}\text{-to}\text{-le}$ )

**lemma cs-path-swap:** **assumes**  $\langle is\text{-path } \pi \rangle \langle is\text{-path } \pi' \rangle \langle \pi =_k \pi' \rangle$  **shows**  $\langle cs^\pi k = cs^{\pi'} k \rangle$  **using**  $assms(1,3)$   
**proof** ( $induction\ \langle \pi \rangle \langle k \rangle\ rule:cs\text{-induct},cases$ )

**case** ( $cs\ \pi\ k$ )  
**let**  $\langle ?l \rangle = \langle (THE\ l.\ k\ icd^\pi \rightarrow l) \rangle$   
**assume**  $*$ :  $\langle \exists l.\ k\ icd^\pi \rightarrow l \rangle$   
**have**  $kicd: \langle k\ icd^\pi \rightarrow ?l \rangle$  **by** ( $metis\ * \ icd\text{-is}\text{-the}\text{-icd}$ )  
**hence**  $\langle ?l < k \rangle$  **unfolding**  $is\text{-cdi}\text{-def}[of\ \langle k \rangle \langle \pi \rangle \langle ?l \rangle]$   $is\text{-icdi}\text{-def}[of\ \langle k \rangle \langle \pi \rangle \langle ?l \rangle]$  **by auto**  
**hence**  $\langle \forall i \leq ?l.\ \pi\ i = \pi' i \rangle$  **using**  $cs(2,3)$  **unfolding**  $eq\text{-up}\text{-to}\text{-def}$  **by auto**  
**hence**  $csl: \langle cs^\pi ?l = cs^{\pi'} ?l \rangle$  **using**  $cs(1,2) \ * \ unfolding\ eq\text{-up}\text{-to}\text{-def}$  **by auto**  
**have**  $kicd: \langle k\ icd^\pi \rightarrow ?l \rangle$  **by** ( $metis\ * \ icd\text{-is}\text{-the}\text{-icd}$ )  
**hence**  $csk: \langle cs^\pi k = cs^\pi ?l @ [\pi\ k] \rangle$  **using**  $kicd$  **by auto**  
**have**  $kicd': \langle k\ icd^{\pi'} \rightarrow ?l \rangle$  **using**  $kicd\ icdi\text{-path}\text{-swap}[OF\ assms(2) - cs(3)]$  **by simp**  
**hence**  $\langle ?l = (THE\ l.\ k\ icd^{\pi'} \rightarrow l) \rangle$  **by** ( $metis\ icd\text{-is}\text{-the}\text{-icd}$ )  
**hence**  $csk': \langle cs^{\pi'} k = cs^{\pi'} ?l @ [\pi' k] \rangle$  **using**  $kicd'$  **by auto**  
**have**  $\langle \pi' k = \pi k \rangle$  **using**  $cs(3)$  **unfolding**  $eq\text{-up}\text{-to}\text{-def}$  **by auto**  
**with**  $csl\ csk\ csk'$   
**show**  $\langle ?case \rangle$  **by auto**

**next**

**case** ( $cs\ \pi\ k$ )  
**assume**  $*$ :  $\langle \neg (\exists l.\ k\ icd^\pi \rightarrow l) \rangle$   
**hence**  $csk: \langle cs^\pi k = [\pi\ k] \rangle$  **by auto**  
**have**  $\langle \neg (\exists l.\ k\ icd^{\pi'} \rightarrow l) \rangle$  **apply** ( $rule\ ccontr$ ) **using**  $* \ icdi\text{-path}\text{-swap}\text{-le}[OF\ cs(2)\ \neg,\ of\ \langle k \rangle \langle \pi' \rangle]$   $cs(3)$  **by**  
( $metis\ eq\text{-up}\text{-to}\text{-sym}\ le\text{-refl}$ )  
**hence**  $csk': \langle cs^{\pi'} k = [\pi' k] \rangle$  **by auto**  
**with**  $csk$  **show**  $\langle ?case \rangle$  **using**  $cs(3)\ eq\text{-up}\text{-to}\text{-apply}$  **by auto**  
**qed**

**lemma cs-path-swap-le:** **assumes**  $\langle is\text{-path } \pi \rangle \langle is\text{-path } \pi' \rangle \langle \pi =_n \pi' \rangle \langle k \leq n \rangle$  **shows**  $\langle cs^\pi k = cs^{\pi'} k \rangle$  **by**  
( $metis\ assms\ cs\text{-path}\text{-swap}\ eq\text{-up}\text{-to}\text{-le}$ )

**lemma cs-path-swap-cd:** **assumes**  $\langle is\text{-path } \pi \rangle$  **and**  $\langle is\text{-path } \pi' \rangle$  **and**  $\langle cs^\pi n = cs^{\pi'} n' \rangle$  **and**  $\langle n\ cd^\pi \rightarrow k \rangle$   
**obtains**  $k'$  **where**  $\langle n'\ cd^{\pi'} \rightarrow k' \rangle$  **and**  $\langle cs^\pi k = cs^{\pi'} k' \rangle$   
**proof** –

**from**  $cd\text{-in}\text{-}cs[OF\ assms(4)]$   
**obtain**  $ns$  **where**  $*$ :  $\langle cs^\pi n = cs^\pi k @ ns @ [\pi n] \rangle$  **by** *blast*  
**obtain**  $xs\ x$  **where**  $csk$ :  $\langle cs^\pi k = xs @ [x] \rangle$  **by** (*metis cs-not-nil rev-exhaust*)  
**have**  $\langle \pi n = \pi' n' \rangle$  **using**  $assms(3)$  *last-cs* **by** *metis*  
**hence**  $*$ :  $\langle cs^{\pi'} n' = xs @ [x] @ ns @ [\pi' n'] \rangle$  **using**  $*$   $assms(3)$   $csk$  **by** *auto*  
**from**  $cs\text{-}split[OF\ **]$   
**obtain**  $k'$  **where**  $\langle cs^{\pi'} k' = xs @ [x] \rangle$   $\langle n' cd^{\pi'} \rightarrow k' \rangle$  **by** *blast*  
**thus**  $\langle thesis \rangle$  **using** *that csk* **by** *auto*

qed

**lemma** *path-ipd-swap*: **assumes**  $\langle is\text{-}path\ \pi \rangle$   $\langle \pi k \neq return \rangle$   $\langle k < n \rangle$   
**obtains**  $\pi' m$  **where**  $\langle is\text{-}path\ \pi' \rangle$   $\langle \pi =_n\ \pi' \rangle$   $\langle k < m \rangle$   $\langle \pi' m = ipd(\pi' k) \rangle$   $\langle \forall l \in \{k..<m\}. \pi' l \neq ipd(\pi' k) \rangle$   
**proof** –  
**obtain**  $\pi' r$  **where**  $*$ :  $\langle \pi' 0 = \pi n \rangle$   $\langle is\text{-}path\ \pi' \rangle$   $\langle \pi' r = return \rangle$  **by** (*metis assms(1) path-nodes reaching-ret*)  
**let**  $\langle ?\pi \rangle = \langle \pi @^n\ \pi' \rangle$   
**have**  $path$ :  $\langle is\text{-}path\ ?\pi \rangle$  **and**  $ret$ :  $\langle ?\pi (n + r) = return \rangle$  **and**  $equpto$ :  $\langle ?\pi =_n\ \pi \rangle$  **using**  $assms\ path\text{-}cons\ *$   
*path-append-eq-up-to* **by** *auto*  
**have**  $\pi k$ :  $\langle ?\pi k = \pi k \rangle$  **by** (*metis assms(3) less-imp-le-nat path-append-def*)  
**obtain**  $j$  **where**  $j$ :  $\langle k < j \wedge j \leq (n + r) \wedge ?\pi j = ipd(\pi k) \rangle$  (**is**  $\langle ?P\ j \rangle$ ) **by** (*metis \pi k assms(2) path path-ret-ipd ret*)  
**define**  $m$  **where**  $m$ :  $\langle m \equiv LEAST\ m . ?P\ m \rangle$   
**have**  $Pm$ :  $\langle ?P\ m \rangle$  **using** *LeastI*[of  $\langle ?P \rangle$ ]  $\langle j \rangle$   $j\ m$  **by** *auto*  
**hence**  $km$ :  $\langle k < m \rangle$   $\langle m \leq (n + r) \rangle$   $\langle ?\pi m = ipd(\pi k) \rangle$  **by** *auto*  
**have**  $le$ :  $\langle \bigwedge l. ?\pi l \implies m \leq l \rangle$  **using** *Least-le*[of  $\langle ?P \rangle$ ]  $m$  **by** *blast*  
**have**  $\pi knipd$ :  $\langle ?\pi k \neq ipd(\pi k) \rangle$  **by** (*metis \pi k assms(1) assms(2) ipd-not-self path-nodes*)  
**have**  $nipd'$ :  $\langle \bigwedge l. k < l \implies l < m \implies ?\pi l \neq ipd(\pi k) \rangle$  **apply** (*rule ccontr*) **using**  $le\ km(2)$  **by** *force*  
**have**  $\langle \forall l \in \{k..<m\}. ?\pi l \neq ipd(\pi k) \rangle$  **using**  $\pi knipd\ nipd'$  **by** (*auto, metis le-eq-less-or-eq, metis le-eq-less-or-eq*)  
**thus**  $\langle thesis \rangle$  **using** *that* **by** (*metis \pi k eq-up-to-sym km(1) km(3) path path-append-eq-up-to*)

qed

**lemma** *cs-sorted-list-of-cd'*:  $\langle cs^\pi k = map\ \pi (sorted\text{-}list\text{-}of\text{-}set\ \{ i . k\ cd^\pi \rightarrow i \}) @ [\pi k] \rangle$   
**proof** (*induction*  $\langle \pi \rangle$   $\langle k \rangle$  *rule: cs.induct, cases*)  
**case** ( $1\ \pi\ k$ )  
**assume**  $\langle \exists j. k\ icd^\pi \rightarrow j \rangle$   
**then obtain**  $j$  **where**  $j$ :  $k\ icd^\pi \rightarrow j ..$   
**hence**  $csj$ :  $\langle cs^\pi j = map\ \pi (sorted\text{-}list\text{-}of\text{-}set\ \{ i . j\ cd^\pi \rightarrow i \}) @ [\pi j] \rangle$  **by** (*metis 1.IH icd-is-the-icd*)  
**have**  $\langle \{ i . k\ cd^\pi \rightarrow i \} = insert\ j\ \{ i . j\ cd^\pi \rightarrow i \} \rangle$  **using** *cdi-is-cd-icdi*[OF  $j$ ] **by** *auto*  
**moreover**  
**have**  $f$ :  $\langle finite\ \{ i . j\ cd^\pi \rightarrow i \} \rangle$  **unfolding** *is-cdi-def* **by** *auto*  
**moreover**  
**have**  $\langle j \notin \{ i . j\ cd^\pi \rightarrow i \} \rangle$  **unfolding** *is-cdi-def* **by** *auto*  
**ultimately**  
**have**  $\langle sorted\text{-}list\text{-}of\text{-}set\ \{ i . k\ cd^\pi \rightarrow i \} = insort\ j (sorted\text{-}list\text{-}of\text{-}set\ \{ i . j\ cd^\pi \rightarrow i \}) \rangle$  **using** *sorted-list-of-set-insert*  
**by** *auto*  
**moreover**  
**have**  $\langle \forall x \in \{ i . j\ cd^\pi \rightarrow i \}. x < j \rangle$  **unfolding** *is-cdi-def* **by** *auto*  
**hence**  $\langle \forall x \in set (sorted\text{-}list\text{-}of\text{-}set\ \{ i . j\ cd^\pi \rightarrow i \}). x < j \rangle$  **by** (*simp add: f*)  
**ultimately**  
**have**  $\langle sorted\text{-}list\text{-}of\text{-}set\ \{ i . k\ cd^\pi \rightarrow i \} = (sorted\text{-}list\text{-}of\text{-}set\ \{ i . j\ cd^\pi \rightarrow i \}) @ [j] \rangle$  **using** *insort-greater* **by**  
*auto*  
**hence**  $\langle cs^\pi j = map\ \pi (sorted\text{-}list\text{-}of\text{-}set\ \{ i . k\ cd^\pi \rightarrow i \}) \rangle$  **using**  $csj$  **by** *auto*  
**thus**  $\langle ?case \rangle$  **by** (*metis icd-cs j*)

**next**  
**case** ( $1\ \pi\ k$ )  
**assume**  $*$ :  $\langle \neg (\exists j. k\ icd^\pi \rightarrow j) \rangle$

**hence**  $\langle cs^\pi k = [\pi k] \rangle$  **by** (*metis cs-cases*)  
**moreover**  
**have**  $\langle \{ i . k \text{ cd}^\pi \rightarrow i \} = \{ \} \rangle$  **by** (*auto, metis \* excd-impl-exicd*)  
**ultimately**  
**show**  $\langle ?case \rangle$  **by** (*metis append-Nil list.simps(8) sorted-list-of-set-empty*)  
**qed**

**lemma cs-sorted-list-of-cd**:  $\langle cs^\pi k = \text{map } \pi (\text{sorted-list-of-set } (\{ i . k \text{ cd}^\pi \rightarrow i \} \cup \{ k \})) \rangle$  **proof** –  
**have** *le*:  $\langle \forall x \in \{ i . k \text{ cd}^\pi \rightarrow i \}. \forall y \in \{ k \}. x < y \rangle$  **unfolding** *is-cdi-def* **by** *auto*  
**have** *fin*:  $\langle \text{finite } \{ i . k \text{ cd}^\pi \rightarrow i \} \rangle$   $\langle \text{finite } \{ k \} \rangle$  **unfolding** *is-cdi-def* **by** *auto*  
**show**  $\langle ?thesis \rangle$  **unfolding** *cs-sorted-list-of-cd* [of  $\langle \pi \rangle$   $\langle k \rangle$ ] *sorted-list-of-set-append* [OF *fin le*] **by** *auto*  
**qed**

**lemma cs-not-ipd**: **assumes**  $\langle k \text{ cd}^\pi \rightarrow j \wedge \text{ipd } (\pi j) \neq \text{ipd } (\pi k) \rangle$  (**is**  $\langle ?Q j \rangle$ )  
**shows**  $\langle cs^\pi (\text{GREATEST } j. k \text{ cd}^\pi \rightarrow j \wedge \text{ipd } (\pi j) \neq \text{ipd } (\pi k)) = [n \leftarrow cs^\pi k . \text{ipd } n \neq \text{ipd } (\pi k)] \rangle$   
(**is**  $\langle cs^\pi ?j = \text{filter } ?P \rightarrow \rangle$ )  
**proof** –

**have** *csk*:  $\langle cs^\pi k = \text{map } \pi (\text{sorted-list-of-set } (\{ i . k \text{ cd}^\pi \rightarrow i \} \cup \{ k \})) \rangle$  **by** (*metis cs-sorted-list-of-cd*)  
**have** *csj*:  $\langle cs^\pi ?j = \text{map } \pi (\text{sorted-list-of-set } (\{ i . ?j \text{ cd}^\pi \rightarrow i \} \cup \{ ?j \})) \rangle$  **by** (*metis cs-sorted-list-of-cd*)

**have** *bound*:  $\langle \forall j. k \text{ cd}^\pi \rightarrow j \wedge \text{ipd } (\pi j) \neq \text{ipd } (\pi k) \longrightarrow j \leq k \rangle$  **unfolding** *is-cdi-def* **by** *simp*

**have** *kcdj*:  $\langle k \text{ cd}^\pi \rightarrow ?j \rangle$  **and** *ipd'*:  $\langle \text{ipd } (\pi ?j) \neq \text{ipd } (\pi k) \rangle$  **using** *GreatestI-nat* [of  $\langle ?Q \rangle$   $\langle j \rangle$   $\langle k \rangle$ , OF *assms*]  
*bound* **by** *auto*

**have** *greatest*:  $\langle \bigwedge j. k \text{ cd}^\pi \rightarrow j \implies \text{ipd } (\pi j) \neq \text{ipd } (\pi k) \implies j \leq ?j \rangle$  **using** *Greatest-le-nat* [of  $\langle ?Q \rangle$  -  $\langle k \rangle$ ]  
*bound* **by** *auto*

**have** *less-not-ipdk*:  $\langle \bigwedge j. k \text{ cd}^\pi \rightarrow j \implies j < ?j \implies \text{ipd } (\pi j) \neq \text{ipd } (\pi k) \rangle$  **by** (*metis (lifting) ipd' kcdj same-ipd-stable*)

**hence** *le-not-ipdk*:  $\langle \bigwedge j. k \text{ cd}^\pi \rightarrow j \implies j \leq ?j \implies \text{ipd } (\pi j) \neq \text{ipd } (\pi k) \rangle$  **using** *kcdj ipd'* **by** (*case-tac*  $\langle j = ?j \rangle$ , *auto*)

**have** *\**:  $\langle \{ j \in \{ i . k \text{ cd}^\pi \rightarrow i \} \cup \{ k \}. ?P (\pi j) \} = \text{insert } ?j \{ i . ?j \text{ cd}^\pi \rightarrow i \} \rangle$   
**apply** *auto*

**apply** (*metis (lifting, no-types) greatest cr-wn'' kcdj le-antisym le-refl*)

**apply** (*metis kcdj*)

**apply** (*metis ipd'*)

**apply** (*metis (full-types) cd-trans kcdj*)

**apply** (*subgoal-tac*  $\langle k \text{ cd}^\pi \rightarrow x \rangle$ )

**apply** (*metis (lifting, no-types) is-cdi-def less-not-ipdk*)

**by** (*metis (full-types) cd-trans kcdj*)

**have**  $\langle \text{finite } (\{ i . k \text{ cd}^\pi \rightarrow i \} \cup \{ k \}) \rangle$  **unfolding** *is-cdi-def* **by** *auto*

**note** *filter-sorted-list-of-set* [OF *this*, of  $\langle ?P \circ \pi \rangle$ ]

**hence**  $\langle [n \leftarrow cs^\pi k . \text{ipd } n \neq \text{ipd } (\pi k)] = \text{map } \pi (\text{sorted-list-of-set } \{ j \in \{ i . k \text{ cd}^\pi \rightarrow i \} \cup \{ k \}. ?P (\pi j) \}) \rangle$

**unfolding** *csk filter-map* **by** *auto*

**also**

**have**  $\langle \dots = \text{map } \pi (\text{sorted-list-of-set } (\text{insert } ?j \{ i . ?j \text{ cd}^\pi \rightarrow i \})) \rangle$  **unfolding** *\** **by** *auto*

**also**

**have**  $\langle \dots = cs^\pi ?j \rangle$  **using** *csj* **by** *auto*

**finally**

**show**  $\langle ?thesis \rangle$  **by** *metis*

**qed**

**lemma cs-ipd**: **assumes** *ipd*:  $\langle \pi m = \text{ipd } (\pi k) \rangle$   $\langle \forall n \in \{ k..<m \}. \pi n \neq \text{ipd } (\pi k) \rangle$

**and** *km*:  $\langle k < m \rangle$  **shows**  $\langle cs^\pi m = [n \leftarrow cs^\pi k . \text{ipd } n \neq \pi m] @ [\pi m] \rangle$

**proof** *cases*

**assume**  $\langle \exists j. m \text{ icd}^\pi \rightarrow j \rangle$

**then obtain  $j$  where  $jicd: \langle m \text{ icd}^\pi \rightarrow j \rangle$  by *blast***  
**hence  $*$ :  $\langle cs^\pi m = cs^\pi j @ [\pi m] \rangle$  by *(metis icd-cs)***  
**have  $j$ :  $\langle j = (\text{GREATEST } j. k \text{ cd}^\pi \rightarrow j \wedge \text{ipd } (\pi j) \neq \pi m) \rangle$  using *jicd assms ipd-icd-greatest-cd-not-ipd* by *blast***  
**moreover**  
**have  $\langle \text{ipd } (\pi j) \neq \text{ipd } (\pi k) \rangle$  by *(metis is-cdi-def is-icdi-def is-ipd-def cd-not-pd ipd(1) ipd-is-ipd jicd path-nodes less-imp-le term-path-stable)***  
**moreover**  
**have  $\langle k \text{ cd}^\pi \rightarrow j \rangle$  unfolding  $j$  by *(metis (lifting, no-types) assms(3) cd-ipd-is-cd icd-imp-cd ipd(1) ipd(2) j jicd)***  
**ultimately**  
**have  $\langle cs^\pi j = [n \leftarrow cs^\pi k . \text{ipd } n \neq \pi m] \rangle$  using *cs-not-ipd[of <k> <pi> <j>] ipd(1)* by *metis***  
**thus  $\langle ?thesis \rangle$  using  $*$  by *metis***  
**next**  
**assume *noicd*:  $\langle \neg (\exists j. m \text{ icd}^\pi \rightarrow j) \rangle$**   
**hence *csm*:  $\langle cs^\pi m = [\pi m] \rangle$  by *auto***  
**have  $\langle \bigwedge j. k \text{ cd}^\pi \rightarrow j \implies \text{ipd}(\pi j) = \pi m \rangle$  using *cd-is-cd-ipd[OF km ipd]* by *(metis excd-impl-excd noicd)***  
**hence  $*$ :  $\langle \{j \in \{i. k \text{ cd}^\pi \rightarrow i\} \cup \{k\}. \text{ipd } (\pi j) \neq \pi m\} = \{\} \rangle$  using *ipd(1)* by *auto***  
**have  $**$ :  $\langle (\lambda n. \text{ipd } n \neq \pi m) o \pi = (\lambda n. \text{ipd } (\pi n) \neq \pi m) \rangle$  by *auto***  
**have *fin*:  $\langle \text{finite } (\{i. k \text{ cd}^\pi \rightarrow i\} \cup \{k\}) \rangle$  unfolding *is-cdi-def* by *auto***  
**note *csk* = *cs-sorted-list-of-cd*[of  $\langle \pi \rangle \langle k \rangle$ ]**  
**hence  $\langle [n \leftarrow cs^\pi k . \text{ipd } n \neq \pi m] = [n \leftarrow (\text{map } \pi (\text{sorted-list-of-set } (\{i. k \text{ cd}^\pi \rightarrow i\} \cup \{k\})))] . \text{ipd } n \neq \pi m \rangle$**   
**by *simp***  
**also**  
**have  $\langle \dots = \text{map } \pi [n \leftarrow \text{sorted-list-of-set } (\{i. k \text{ cd}^\pi \rightarrow i\} \cup \{k\}). \text{ipd } (\pi n) \neq \pi m] \rangle$  by *(auto simp add: filter-map \*\*)***  
**also**  
**have  $\langle \dots = [] \rangle$  unfolding  $*$  *filter-sorted-list-of-set*[OF *fin*, of  $\langle \lambda n. \text{ipd } (\pi n) \neq \pi m \rangle$ ] by *auto***  
**finally**  
**show  $\langle ?thesis \rangle$  using *csm* by *(metis append-Nil)***  
**qed**

**lemma *converged-ipd-same-icd*: assumes *path*:  $\langle \text{is-path } \pi \rangle \langle \text{is-path } \pi' \rangle$  and *converge*:  $\langle l < m \rangle \langle cs^\pi m = cs^{\pi'} m' \rangle$**   
**and *csk*:  $\langle cs^\pi k = cs^{\pi'} k' \rangle$  and *icd*:  $\langle l \text{ icd}^\pi \rightarrow k \rangle$  and *suc*:  $\langle \pi (\text{Suc } k) = \pi' (\text{Suc } k') \rangle$**   
**and *ipd*:  $\langle \pi' m' = \text{ipd } (\pi k) \rangle \langle \forall n \in \{k'..<m'\}. \pi' n \neq \text{ipd } (\pi k) \rangle$**   
**shows  $\langle \exists l'. cs^\pi l = cs^{\pi'} l' \rangle$**   
**proof cases**  
**assume *l*:  $\langle l = \text{Suc } k \rangle$**   
**hence  $\langle \text{Suc } k \text{ cd}^\pi \rightarrow k \rangle$  using *icd* by *(metis is-icdi-def)***  
**hence  $\langle \pi (\text{Suc } k) \neq \text{ipd } (\pi k) \rangle$  unfolding *is-cdi-def* by *auto***  
**hence  $\langle \pi' (\text{Suc } k') \neq \text{ipd } (\pi' k') \rangle$  by *(metis csk last-cs suc)***  
**moreover**  
**have  $\langle \pi' (\text{Suc } k') \neq \text{return} \rangle$  by *(metis <Suc k cd<sup>π</sup> → k> ret-no-cd suc)***  
**ultimately**  
**have  $\langle \text{Suc } k' \text{ cd}^{\pi'} \rightarrow k' \rangle$  unfolding *is-cdi-def* using *path(2)* apply *auto***  
**by *(metis ipd-not-self le-Suc-eq le-antisym path-nodes term-path-stable)***  
**hence  $\langle \text{Suc } k' \text{ icd}^{\pi'} \rightarrow k' \rangle$  unfolding *is-icdi-def* using *path(2)* by *fastforce***  
**hence  $\langle cs^{\pi'} (\text{Suc } k') = cs^{\pi'} k' @ [\pi' (\text{Suc } k')] \rangle$  using *icd-cs* by *auto***  
**moreover**  
**have  $\langle cs^\pi l = cs^\pi k @ [\pi l] \rangle$  using *icd icd-cs* by *auto***  
**ultimately**  
**have  $\langle cs^\pi l = cs^{\pi'} (\text{Suc } k') \rangle$  by *(metis csk l suc)***  
**thus  $\langle ?thesis \rangle$  by *blast***  
**next**

**assume**  $nsuck$ :  $\langle l \neq Suc\ k \rangle$   
**have**  $kk'$ [*simp*]:  $\langle \pi' k' = \pi\ k \rangle$  **by** (*metis csk last-cs*)  
**have**  $kl$ :  $\langle k < l \rangle$  **using** *icd unfolding is-icdi-def is-cdi-def* **by** *auto*  
**hence**  $skl$ :  $\langle Suc\ k < l \rangle$  **by** (*metis Suc-lessI nsuck*)  
**hence**  $lpd$ :  $\langle \pi\ l\ pd \rightarrow \pi\ (Suc\ k) \rangle$  **using** *icd icd-pd-intermediate* **by** *auto*  
**have**  $km$ :  $\langle k < m \rangle$  **by** (*metis converge(1) kl order.strict-trans*)  
**have**  $lcd$ :  $\langle l\ cd^\pi \rightarrow k \rangle$  **using** *icd is-icdi-def* **by** *auto*  
**hence**  $ipdk-pdl$ :  $\langle ipd\ (\pi\ k)\ pd \rightarrow (\pi\ l) \rangle$  **by** (*metis ipd-pd-cd*)  
**have**  $*$ :  $\langle ipd\ (\pi\ k) \in nodes \rangle$  **by** (*metis ipdk-pdl pd-node1*)  
**have**  $nretk$ :  $\langle \pi\ k \neq return \rangle$  **by** (*metis kl lcd path(1) ret-no-cd term-path-stable less-imp-le*)  
**have**  $**$ :  $\langle \neg (\pi\ l)\ pd \rightarrow ipd\ (\pi\ k) \rangle$  **proof**  
  **assume**  $a$ :  $\langle \pi\ l\ pd \rightarrow ipd\ (\pi\ k) \rangle$   
  **hence**  $\langle \pi\ l\ pd \rightarrow (\pi\ k) \rangle$  **by** (*metis is-ipd-def  $\langle k < l \rangle ipd-is-ipd ipdk-pdl path(1) path-nodes pd-antisym term-path-stable less-imp-le$* )  
  **moreover**  
  **have**  $\langle \pi\ l \neq (\pi\ k) \rangle$  **by** (*metis \* a ipd-not-self ipdk-pdl lcd pd-antisym ret-no-cd*)  
  **ultimately**  
  **show**  $\langle False \rangle$  **using** *lcd cd-not-pd* **by** *auto*  
**qed**

**have**  $km'$ :  $\langle k' < m' \rangle$  **using** *cs-order[OF path csk converge(2) nretk km]* .

**obtain**  $\pi''\ n''$  **where**  $path''$ :  $\langle is-path\ \pi'' \rangle$  **and**  $\pi''0$ :  $\langle \pi''\ 0 = ipd\ (\pi\ k) \rangle$  **and**  $\pi''n$ :  $\langle \pi''\ n'' = return \rangle$  **and**  $not\pi l$ :  $\langle \forall\ i \leq n''.\ \pi''\ i \neq \pi\ l \rangle$  **using** *no-pd-path[OF \* \*\*]* .

**let**  $\langle ?\pi' \rangle = \langle (\pi' @^{m'} \pi'') \ll Suc\ k' \rangle$

**have**  $\langle is-path\ ?\pi' \rangle$  **by** (*metis  $\pi''0 ipd(1) path'' path(2) path-cons path-path-shift$* )

**moreover**

**have**  $\langle ?\pi'\ 0 = \pi\ (Suc\ k) \rangle$  **using** *km' suc* **by** *auto*

**moreover**

**have**  $\langle ?\pi'\ (m' - Suc\ k' + n'') = return \rangle$  **using**  $\pi''n\ km'\ \pi''0\ ipd(1)$  **by** *auto*

**ultimately**

**obtain**  $l''$  **where**  $l''$ :  $\langle l'' \leq m' - Suc\ k' + n'' \rangle$   $\langle ?\pi'\ l'' = \pi\ l \rangle$  **using** *lpd unfolding is-pd-def* **by** *blast*

**have**  $l''m$ :  $\langle l'' \leq m' - Suc\ k' \rangle$  **apply** (*rule ccontr*) **using**  $l''\ not\pi l\ km'$  **by** (*cases  $\langle Suc\ (k' + l'') \leq m' \rangle$ , auto*)

**let**  $\langle ?l' \rangle = \langle Suc\ (k' + l'') \rangle$

**have**  $lm'$ :  $\langle ?l' \leq m' \rangle$  **using**  $l''m\ km'$  **by** *auto*

— Now we have found our desired  $l'$

**have**  $1$ :  $\langle \pi'\ ?l' = \pi\ l \rangle$  **using**  $l''\ l''m\ lm'$  **by** *auto*

**have**  $2$ :  $\langle k' < ?l' \rangle$  **by** *simp*

**have**  $3$ :  $\langle ?l' < m' \rangle$  **apply** (*rule ccontr*) **using**  $lm'$  **by** (*simp, metis \*\* 1 ipd(1) ipdk-pdl*)

— Need the least such  $l'$

**let**  $\langle ?P \rangle = \langle \lambda\ l'.\ \pi'\ l' = \pi\ l \wedge k' < l' \wedge l' < m' \rangle$

**have**  $*$ :  $\langle ?P\ ?l' \rangle$  **using**  $1\ 2\ 3$  **by** *blast*

**define**  $l'$  **where**  $l'$ :  $\langle l' == LEAST\ l'.\ ?P\ l' \rangle$

**have**  $\pi l'$ :  $\langle \pi'\ l' = \pi\ l \rangle$  **using**  $l'\ 1\ 2\ 3\ LeastI[of\ \langle ?P \rangle]$  **by** *blast*

**have**  $kl'$ :  $\langle k' < l' \rangle$  **using**  $l'\ 1\ 2\ 3\ LeastI[of\ \langle ?P \rangle]$  **by** *blast*

**have**  $lm'$ :  $\langle l' < m' \rangle$  **using**  $l'\ 1\ 2\ 3\ LeastI[of\ \langle ?P \rangle]$  **by** *blast*

**have**  $nretl'$ :  $\langle \pi'\ l' \neq return \rangle$  **by** (*metis  $\pi''n\ \pi l'\ le-refl\ not\pi l$* )

**have**  $nipd'$ :  $\langle \forall\ j \in \{k'..l'\}.\ \pi'\ j \neq ipd\ (\pi'\ k') \rangle$  **using**  $lm'\ kk'\ ipd(2)\ kl'$  **by** *force*



**have**  $lcd'$ :  $\langle l' cd^{\pi'} \rightarrow k' \rangle$  **by** (*metis is-cdi-def kl' nipd' nretl' path(2)*)

**have**  $licd$ :  $\langle l' icd^{\pi'} \rightarrow k' \rangle$  **proof** –

**have**  $\langle \forall m \in \{k' <..<l'\}. \neg l' cd^{\pi'} \rightarrow m \rangle$  **proof** (*rule ccontr*)

**assume**  $\langle \neg (\forall m \in \{k' <..<l'\}. \neg l' cd^{\pi'} \rightarrow m) \rangle$

**then obtain**  $j'$  **where**  $kj'$ :  $\langle k' < j' \rangle$  **and**  $jl'$ :  $\langle j' < l' \rangle$  **and**  $lcdj'$ :  $\langle l' cd^{\pi'} \rightarrow j' \rangle$  **by** *force*

**have**  $jm'$ :  $\langle j' < m' \rangle$  **by** (*metis jl' lm' order.strict-trans*)

**have**  $\langle \pi' j' \neq \pi l' \rangle$  **apply** (*rule ccontr*) **using**  $l' kj' jm' jl'$  *Least-le[of  $\langle ?P \rangle \langle j' \rangle$ ]* **by** *auto*

**hence**  $\langle \neg \pi' l' pd \rightarrow \pi' j' \rangle$  **using**  $cd\text{-not-pd } lcdj' \pi l'$  **by** *metis*

**moreover have**  $\langle \pi' j' \in nodes \rangle$  **using**  $path(2)$  *path-nodes* **by** *auto*

**ultimately**

**obtain**  $\pi_1 n_1$  **where**  $path_1$ :  $\langle is\text{-path } \pi_1 \rangle$  **and**  $\pi 0_1$ :  $\langle \pi_1 0 = \pi' j' \rangle$  **and**  $\pi n_1$ :  $\langle \pi_1 n_1 = return \rangle$  **and**  $nl'$ :  $\langle \forall l \leq n_1. \pi_1 l \neq \pi' l' \rangle$  **unfolding** *is-pd-def* **by** *blast*

**let**  $\langle ?\pi'' \rangle = \langle \pi' @^{j'} \pi_1 \rangle \ll Suc\ k'$

**have**  $\langle is\text{-path } ?\pi'' \rangle$  **by** (*metis  $\pi 0_1$  path(2) path\_1 path-cons path-path-shift*)

**moreover**

**have**  $\langle ?\pi'' 0 = \pi (Suc\ k) \rangle$  **by** (*simp, metis kj' less-eq-Suc-le suc*)

**moreover**

**have**  $kj'$ :  $\langle Suc\ k' \leq j' \rangle$  **by** (*metis kj' less-eq-Suc-le*)

**hence**  $\langle ?\pi'' (j' - Suc\ k' + n_1) = return \rangle$  **by** (*simp, metis  $\pi 0_1 \pi n_1$* )

**ultimately**

**obtain**  $l''$  **where**  $*$ :  $\langle ?\pi'' l'' = \pi l' \rangle$  **and**  $**$ :  $\langle l'' \leq j' - Suc\ k' + n_1 \rangle$  **using** *lpd is-pd-def* **by** *blast*

**show**  $\langle False \rangle$  **proof** (*cases*)

**assume**  $a$ :  $\langle l'' \leq j' - Suc\ k' \rangle$

**hence**  $\langle \pi' (l'' + Suc\ k') = \pi l' \rangle$  **using**  $*$   $kj'$  **by** (*simp, metis Nat.le-diff-conv2 add-Suc diff-add-inverse le-add1 le-add-diff-inverse2*)

**moreover**

**have**  $\langle l'' + Suc\ k' < l' \rangle$  **by** (*metis a jl' add-diff-cancel-right' kj' le-add-diff-inverse less-imp-diff-less ordered-cancel-comm-monoid-diff-class.le-diff-conv2*)

**moreover**

**have**  $\langle l'' + Suc\ k' < m' \rangle$  **by** (*metis Suc-lessD calculation(2) less-trans-Suc lm'*)

**moreover**

**have**  $\langle k' < l'' + Suc\ k' \rangle$  **by** *simp*

**ultimately**

**show**  $\langle False \rangle$  **using** *Least-le[of  $\langle ?P \rangle \langle l'' + Suc\ k' \rangle$ ]*  $l'$  **by** *auto*

**next**

**assume**  $a$ :  $\langle \neg l'' \leq j' - Suc\ k' \rangle$

**hence**  $\langle \neg Suc\ (k' + l'') \leq j' \rangle$  **by** *simp*

**hence**  $\langle \pi_1 (Suc\ (k' + l'') - j') = \pi l' \rangle$  **using**  $*$   $kj'$  **by** *simp*

**moreover**

**have**  $\langle Suc\ (k' + l'') - j' \leq n_1 \rangle$  **using**  $**$   $kj'$  **by** *simp*

**ultimately**

**show**  $\langle False \rangle$  **using**  $nl'$  **by** (*metis  $\pi l'$* )

**qed**

**qed**

**thus**  $\langle ?thesis \rangle$  **unfolding** *is-icdi-def* **using**  $lcd'$   $path(2)$  **by** *simp*

**qed**

**hence**  $\langle cs^{\pi'} l' = cs^{\pi'} k' @ [\pi' l'] \rangle$  **by** (*metis icd-cs*)

**hence**  $\langle cs^{\pi'} l' = cs^{\pi'} l' \rangle$  **by** (*metis  $\pi l'$  csk icd icd-cs*)

**thus**  $\langle ?thesis \rangle$  **by** *metis*

**qed**

**lemma** *converged-same-icd*: **assumes**  $path$ :  $\langle is\text{-path } \pi \rangle$   $\langle is\text{-path } \pi' \rangle$  **and**  $converge$ :  $\langle l < n \rangle$   $\langle cs^{\pi} n = cs^{\pi'} n' \rangle$  **and**  $csk$ :  $\langle cs^{\pi} k = cs^{\pi'} k' \rangle$  **and**  $icd$ :  $\langle l icd^{\pi} \rightarrow k \rangle$  **and**  $suc$ :  $\langle \pi (Suc\ k) = \pi' (Suc\ k') \rangle$

shows  $\langle \exists l'. cs^\pi l = cs^{\pi'} l' \rangle$  **proof** –

**have**  $nret: \langle \pi k \neq return \rangle$  **using** *icd unfolding is-icdi-def is-cdi-def* **using** *term-path-stable less-imp-le* **by** *metis*

**have**  $kl: \langle k < l \rangle$  **using** *icd unfolding is-icdi-def is-cdi-def* **by** *auto*

**have**  $kn: \langle k < n \rangle$  **using** *converge kl* **by** *simp*

**from** *path-ipd-swap[OF path(1) nret kn]*

**obtain**  $\varrho m$  **where**  $path\varrho: \langle is-path \varrho \rangle$  **and**  $\pi\varrho: \langle \pi =_n \varrho \rangle$  **and**  $km: \langle k < m \rangle$  **and**  $ipd: \langle \varrho m = ipd(\varrho k) \rangle$   $\langle \forall l \in \{k..<m\}. \varrho l \neq ipd(\varrho k) \rangle$  .

**have**  $csk1: \langle cs^\varrho k = cs^\pi k \rangle$  **using** *cs-path-swap-le path path\varrho \pi\varrho kn* **by** *auto*

**have**  $suc\varrho: \langle \varrho(Suc k) = \pi(Suc k) \rangle$  **by** (*metis \pi\varrho eq-up-to-def kn less-eq-Suc-le*)

**have**  $nret': \langle \pi' k' \neq return \rangle$  **by** (*metis csk last-cs nret*)

**have**  $kn': \langle k' < n' \rangle$  **using** *cs-order[OF path csk converge(2) nret kn]* .

**from** *path-ipd-swap[OF path(2) nret' kn']*

**obtain**  $\varrho' m'$  **where**  $path\varrho': \langle is-path \varrho' \rangle$  **and**  $\pi\varrho': \langle \pi' =_{n'} \varrho' \rangle$  **and**  $km': \langle k' < m' \rangle$  **and**  $ipd': \langle \varrho' m' = ipd(\varrho' k') \rangle$   $\langle \forall l \in \{k'..<m'\}. \varrho' l \neq ipd(\varrho' k') \rangle$  .

**have**  $csk1': \langle cs^{\varrho'} k' = cs^{\pi'} k' \rangle$  **using** *cs-path-swap-le path path\varrho' \pi\varrho' kn'* **by** *auto*

**have**  $suc\varrho': \langle \varrho'(Suc k') = \pi'(Suc k') \rangle$  **by** (*metis \pi\varrho' eq-up-to-def kn' less-eq-Suc-le*)

**have**  $icd\varrho: \langle l icd^\varrho \rightarrow k \rangle$  **using** *icdi-path-swap-le[OF path\varrho icd \pi\varrho]* **converge** **by** *simp*

**have**  $lm: \langle l < m \rangle$  **using** *ipd(1) icd\varrho km unfolding is-icdi-def is-cdi-def* **by** *auto*

**have**  $csk': \langle cs^\varrho k = cs^{\varrho'} k' \rangle$  **using** *csk1 csk1' csk* **by** *auto*

**hence**  $kk': \langle \varrho' k' = \varrho k \rangle$  **using** *last-cs* **by** *metis*

**have**  $suc': \langle \varrho(Suc k) = \varrho'(Suc k') \rangle$  **using** *suc suc\varrho suc\varrho'* **by** *auto*

**have**  $mm': \langle \varrho' m' = \varrho m \rangle$  **using** *ipd(1) ipd'(1) kk'* **by** *auto*

**from** *cs-ipd[OF ipd km] cs-ipd[OF ipd' km',unfolding mm', folded csk']*

**have**  $csm: \langle cs^\varrho m = cs^{\varrho'} m' \rangle$  **by** *metis*

**from** *converged-ipd-same-icd[OF path\varrho path\varrho' lm csm csk' icd\varrho suc' ipd'[unfolding kk']]*

**obtain**  $l'$  **where**  $csl: \langle cs^\varrho l = cs^{\varrho'} l' \rangle$  **by** *blast*

**have**  $csl\varrho: \langle cs^\pi l = cs^\varrho l \rangle$  **using**  $\pi\varrho$  *converge(1) cs-path-swap-le less-imp-le-nat path(1) path\varrho* **by** *blast*

**have**  $nretl: \langle \varrho l \neq return \rangle$  **by** (*metis icd\varrho icd-imp-cd ret-no-cd*)

**have**  $csn': \langle cs^\varrho n = cs^{\varrho'} n' \rangle$  **using** *converge(2) cs-path-swap path path\varrho path\varrho' \pi\varrho \pi\varrho'* **by** *auto*

**have**  $ln': \langle l' < n' \rangle$  **using** *cs-order[OF path\varrho path\varrho' csl csn' nretl converge(1)]* .

**have**  $csl\varrho': \langle cs^{\pi'} l' = cs^{\varrho'} l' \rangle$  **using** *cs-path-swap-le[OF path(2) path\varrho' \pi\varrho']*  $ln'$  **by** *auto*

**have**  $csl': \langle cs^\pi l = cs^{\pi'} l' \rangle$  **using** *csl\varrho csl\varrho' csl* **by** *auto*

**thus**  $\langle ?thesis \rangle$  **by** *blast*

**qed**

**lemma** *cd-is-cs-less*: **assumes**  $\langle l cd^\pi \rightarrow k \rangle$  **shows**  $\langle cs^\pi k < cs^\pi l \rangle$  **proof** –

**obtain**  $xs$  **where**  $csl: \langle cs^\pi l = cs^\pi k @ xs @[\pi l] \rangle$  **using** *cd-in-cs[OF assms]* **by** *blast*

**hence**  $len: \langle length(cs^\pi k) < length(cs^\pi l) \rangle$  **by** *auto*

have take:  $\langle \text{take } (\text{length } (cs^\pi k)) (cs^\pi l) = cs^\pi k \rangle$  using csl by auto  
 show  $\langle ?thesis \rangle$  using cs-less.intros[OF len take] .

qed

lemma cs-select-id: assumes  $\langle is\text{-path } \pi \rangle \langle \pi k \neq \text{return} \rangle$  shows  $\langle \pi | cs^\pi k = k \rangle$  (is  $\langle ?k = k \rangle$ ) proof –  
 have \*:  $\langle \bigwedge i. cs^\pi i = cs^\pi k \implies i = k \rangle$  using cs-inj[OF assms] by metis  
 hence  $\langle cs^\pi ?k = cs^\pi k \rangle$  unfolding cs-select-def using theI[of  $\langle \lambda i. cs^\pi i = cs^\pi k \rangle \langle k \rangle$ ] by auto  
 thus  $\langle ?k = k \rangle$  using \* by auto

qed

lemma cs-single-nocd: assumes  $\langle cs^\pi i = [x] \rangle$  shows  $\langle \forall k. \neg i \text{ cd}^\pi \rightarrow k \rangle$  proof –  
 have  $\langle \neg (\exists k. i \text{ icd}^\pi \rightarrow k) \rangle$  apply (rule ccontr) using assms cs-not-nil by auto  
 hence  $\langle \neg (\exists k. i \text{ cd}^\pi \rightarrow k) \rangle$  by (metis excd-impl-exicd)  
 thus  $\langle ?thesis \rangle$  by blast

qed

lemma cs-single-pd-intermed: assumes  $\langle is\text{-path } \pi \rangle \langle cs^\pi n = [\pi n] \rangle \langle k \leq n \rangle$  shows  $\langle \pi n \text{ pd} \rightarrow \pi k \rangle$  proof –  
 have  $\langle \forall l. \neg n \text{ icd}^\pi \rightarrow l \rangle$  by (metis assms(2) cs-single-nocd icd-imp-cd)  
 thus  $\langle ?thesis \rangle$  by (metis assms(1) assms(3) no-icd-pd)

qed

lemma cs-first-pd: assumes path:  $\langle is\text{-path } \pi \rangle$  and pd:  $\langle \pi n \text{ pd} \rightarrow \pi 0 \rangle$  and first:  $\langle \forall l < n. \pi l \neq \pi n \rangle$  shows  
 $\langle cs^\pi n = [\pi n] \rangle$   
 by (metis cs-cases first first-pd-no-cd icd-imp-cd path pd)

lemma converged-pd-cs-single: assumes path:  $\langle is\text{-path } \pi \rangle \langle is\text{-path } \pi' \rangle$  and converge:  $\langle l < m \rangle \langle cs^\pi m = cs^{\pi'} m' \rangle$   
 and  $\pi 0$ :  $\langle \pi 0 = \pi' 0 \rangle$  and mpdl:  $\langle \pi m \text{ pd} \rightarrow \pi l \rangle$  and csl:  $\langle cs^\pi l = [\pi l] \rangle$   
 shows  $\langle \exists l'. cs^\pi l = cs^{\pi'} l' \rangle$  proof –  
 have \*:  $\langle \pi l \text{ pd} \rightarrow \pi' 0 \rangle$  using cs-single-pd-intermed[OF path(1) csl]  $\pi 0$ [symmetric] by auto  
 have  $\pi m$ :  $\langle \pi m = \pi' m' \rangle$  by (metis converge(2) last-cs)  
 hence \*\*:  $\langle \pi' m' \text{ pd} \rightarrow \pi l \rangle$  using mpdl by metis

obtain  $l'$  where  $lm'$ :  $\langle l' \leq m' \rangle$  and  $\pi l$ :  $\langle \pi' l' = \pi l \rangle$  (is  $\langle ?P l' \rangle$ ) using path-pd-pd0[OF path(2) \*\* \*] .

let  $\langle ?l \rangle = \langle \text{LEAST } l'. \pi' l' = \pi l \rangle$

have  $\pi l'$ :  $\langle \pi' ?l = \pi l \rangle$  using LeastI[of  $\langle ?P \rangle$ , OF  $\pi l$ ] .

moreover

have  $\langle \forall i < ?l. \pi' i \neq \pi l \rangle$  using Least-le[of  $\langle ?P \rangle$ ] by (metis not-less)

hence  $\langle \forall i < ?l. \pi' i \neq \pi' ?l \rangle$  using  $\pi l'$  by metis

moreover

have  $\langle \pi' ?l \text{ pd} \rightarrow \pi' 0 \rangle$  using \*  $\pi l'$  by metis

ultimately

have  $\langle cs^{\pi'} ?l = [\pi' ?l] \rangle$  using cs-first-pd[OF path(2)] by metis

thus  $\langle ?thesis \rangle$  using csl  $\pi l'$  by metis

qed

lemma converged-cs-single: assumes path:  $\langle is\text{-path } \pi \rangle \langle is\text{-path } \pi' \rangle$  and converge:  $\langle l < m \rangle \langle cs^\pi m = cs^{\pi'} m' \rangle$

and  $\pi 0$ :  $\langle \pi 0 = \pi' 0 \rangle$  and csl:  $\langle cs^\pi l = [\pi l] \rangle$

shows  $\langle \exists l'. cs^\pi l = cs^{\pi'} l' \rangle$  proof cases

assume \*:  $\langle \pi l = \text{return} \rangle$

hence  $\langle \pi m = \text{return} \rangle$  by (metis converge(1) path(1) term-path-stable less-imp-le)

**hence**  $\langle cs^\pi m = [return] \rangle$  **using** *cs-return* **by** *auto*  
**hence**  $\langle cs^{\pi'} m' = [return] \rangle$  **using** *converge* **by** *simp*  
**moreover**  
**have**  $\langle cs^\pi l = [return] \rangle$  **using**  $*$  *cs-return* **by** *auto*  
**ultimately show**  $\langle ?thesis \rangle$  **by** *metis*  
**next**  
**assume** *nret*:  $\langle \pi l \neq return \rangle$   
**have**  $\pi m$ :  $\langle \pi m = \pi' m' \rangle$  **by** (*metis converge(2) last-cs*)  
  
**obtain**  $\pi_1 n$  **where** *path1*:  $\langle is-path \pi_1 \rangle$  **and** *upto*:  $\langle \pi =_m \pi_1 \rangle$  **and**  $\pi n$ :  $\langle \pi_1 n = return \rangle$  **using** *path(1)*  
*path-swap-ret* **by** *blast*  
  
**obtain**  $\pi_1' n'$  **where** *path1'*:  $\langle is-path \pi_1' \rangle$  **and** *upto'*:  $\langle \pi' =_{m'} \pi_1' \rangle$  **and**  $\pi n'$ :  $\langle \pi_1' n' = return \rangle$  **using** *path(2)*  
*path-swap-ret* **by** *blast*  
  
**have**  $\pi 1l$ :  $\langle \pi_1 l = \pi l \rangle$  **using** *upto converge(1)* **by** (*metis eq-up-to-def nat-less-le*)  
  
**have** *cs1l*:  $\langle cs^{\pi_1} l = cs^\pi l \rangle$  **using** *cs-path-swap-le upto path1 path(1) converge(1)* **by** *auto*  
  
**have** *csl1*:  $\langle cs^{\pi_1} l = [\pi_1 l] \rangle$  **by** (*metis pi1 cs1l csl*)  
  
**have** *converge1*:  $\langle cs^{\pi_1} n = cs^{\pi_1'} n' \rangle$  **using**  $\pi n \pi n'$  *cs-return* **by** *auto*  
  
**have** *ln*:  $\langle l < n \rangle$  **using** *nret pi n pi1l term-path-stable[OF path1 pi n]* **by** (*auto, metis linorder-neqE-nat less-imp-le*)  
  
**have**  $\pi 0l$ :  $\langle \pi_1 0 = \pi_1' 0 \rangle$  **using**  $\pi 0$  *eq-up-to-apply[OF upto]* *eq-up-to-apply[OF upto']* **by** *auto*  
  
**have** *pd*:  $\langle \pi_1 n pd \rightarrow \pi_1 l \rangle$  **using**  $\pi n$  **by** (*metis path1 path-nodes return-pd*)  
  
**obtain** *l'* **where** *csl*:  $\langle cs^{\pi_1} l = cs^{\pi_1'} l' \rangle$  **using** *converged-pd-cs-single[OF path1 path1' ln converge1 pi0l pd csl1]* **by** *blast*  
  
**have** *cs1m*:  $\langle cs^{\pi_1} m = cs^\pi m \rangle$  **using** *cs-path-swap upto path1 path(1)* **by** *auto*  
**have** *cs1m'*:  $\langle cs^{\pi_1'} m' = cs^{\pi'} m' \rangle$  **using** *cs-path-swap upto' path1' path(2)* **by** *auto*  
**hence** *converge1*:  $\langle cs^{\pi_1} m = cs^{\pi_1'} m' \rangle$  **using** *converge(2) cs1m* **by** *metis*  
  
**have** *nret1*:  $\langle \pi_1 l \neq return \rangle$  **using** *nret pi1l* **by** *auto*  
  
**have** *lm'*:  $\langle l' < m' \rangle$  **using** *cs-order[OF path1 path1' csl converge1 nret1 converge(1)]* .  
  
**have**  $\langle cs^{\pi'} l' = cs^{\pi_1'} l' \rangle$  **using** *cs-path-swap-le[OF path(2) path1' upto']* *lm'* **by** *auto*  
**moreover**  
**have**  $\langle cs^\pi l = cs^{\pi_1} l \rangle$  **using** *cs-path-swap-le[OF path(1) path1 upto]* *converge(1)* **by** *auto*  
**ultimately**  
**have**  $\langle cs^\pi l = cs^{\pi'} l' \rangle$  **using** *csl* **by** *auto*  
**thus**  $\langle ?thesis \rangle$  **by** *blast*  
**qed**  
  
**lemma** *converged-cd-same-suc*: **assumes** *path*:  $\langle is-path \pi \rangle$   $\langle is-path \pi' \rangle$  **and** *init*:  $\langle \pi 0 = \pi' 0 \rangle$   
**and** *cd-suc*:  $\langle \forall k k'. cs^\pi k = cs^{\pi'} k' \wedge l cd^\pi \rightarrow k \longrightarrow \pi (Suc k) = \pi' (Suc k') \rangle$  **and** *converge*:  $\langle l < m \rangle$   $\langle cs^\pi m = cs^{\pi'} m' \rangle$   
**shows**  $\langle \exists l'. cs^\pi l = cs^{\pi'} l' \rangle$   
**using** *path init cd-suc converge* **proof** (*induction*  $\langle \pi \rangle$   $\langle l \rangle$  *rule*: *cs-induct,cases*)  
**case** (*cs pi l*)

**assume** \*:  $\langle \exists k. l \text{ icd}^\pi \rightarrow k \rangle$   
**let**  $\langle ?k \rangle = \langle \text{THE } k. l \text{ icd}^\pi \rightarrow k \rangle$   
**have**  $\text{icd}$ :  $\langle l \text{ icd}^\pi \rightarrow ?k \rangle$  **by** (*metis* \* *icd-is-the-icd*)  
**hence**  $\text{lcdk}$ :  $\langle l \text{ cd}^\pi \rightarrow ?k \rangle$  **by** (*metis is-icdi-def*)  
**hence**  $\text{kl}$ :  $\langle ?k < l \rangle$  **using** *is-cdi-def* **by** *metis*  
  
**have**  $\langle \bigwedge j. ?k \text{ cd}^\pi \rightarrow j \implies l \text{ cd}^\pi \rightarrow j \rangle$  **using** *icd cd-trans is-icdi-def* **by** *fast*  
**hence**  $\text{suc}'$ :  $\langle \forall j j'. \text{cs}^\pi j = \text{cs}^{\pi'} j' \wedge ?k \text{ cd}^\pi \rightarrow j \longrightarrow \pi (\text{Suc } j) = \pi' (\text{Suc } j') \rangle$  **using** *cs.prem(4)* **by** *blast*  
  
**from** *cs.IH[OF \* cs(2) path(2) cs(4) suc'] cs.prem kl*  
**have**  $\langle \exists k'. \text{cs}^\pi (\text{THE } k. l \text{ icd}^\pi \rightarrow k) = \text{cs}^{\pi'} k' \rangle$  **by** (*metis Suc-lessD less-trans-Suc*)  
**then obtain**  $k'$  **where**  $\text{csk}$ :  $\langle \text{cs}^\pi ?k = \text{cs}^{\pi'} k' \rangle$  **by** *blast*  
  
**have**  $\text{suc2}$ :  $\langle \pi (\text{Suc } ?k) = \pi' (\text{Suc } k') \rangle$  **using** *cs.prem(4) lcdk csk* **by** *auto*  
  
**have**  $\text{km}$ :  $\langle ?k < m \rangle$  **using** *kl cs.prem(5)* **by** *simp*  
  
**from** *converged-same-icd[OF cs(2) path(2) cs.prem(5) cs.prem(6) csk icd suc2]*  
**show**  $\langle ?\text{case} \rangle$  .

**next**  
**case** (*cs*  $\pi$   $l$ )  
**assume**  $\langle \neg (\exists k. l \text{ icd}^\pi \rightarrow k) \rangle$   
**hence**  $\langle \text{cs}^\pi l = [\pi l] \rangle$  **by** *auto*  
**with** *cs converged-cs-single*  
**show**  $\langle ?\text{case} \rangle$  **by** *metis*  
**qed**

**lemma** *converged-cd-diverge*:

**assumes**  $\text{path}$ :  $\langle \text{is-path } \pi \rangle \langle \text{is-path } \pi' \rangle$  **and**  $\text{init}$ :  $\langle \pi 0 = \pi' 0 \rangle$  **and**  $\text{notin}$ :  $\langle \neg (\exists l'. \text{cs}^\pi l = \text{cs}^{\pi'} l') \rangle$  **and**  
 $\text{converge}$ :  $\langle l < m \rangle \langle \text{cs}^\pi m = \text{cs}^{\pi'} m' \rangle$   
**obtains**  $k k'$  **where**  $\langle \text{cs}^\pi k = \text{cs}^{\pi'} k' \rangle \langle l \text{ cd}^\pi \rightarrow k \rangle \langle \pi (\text{Suc } k) \neq \pi' (\text{Suc } k') \rangle$   
**using** *assms converged-cd-same-suc* **by** *blast*

**lemma** *converged-cd-same-suc-return*: **assumes**  $\text{path}$ :  $\langle \text{is-path } \pi \rangle \langle \text{is-path } \pi' \rangle$  **and**  $\pi 0$ :  $\langle \pi 0 = \pi' 0 \rangle$   
**and**  $\text{cd-suc}$ :  $\langle \forall k k'. \text{cs}^\pi k = \text{cs}^{\pi'} k' \wedge l \text{ cd}^\pi \rightarrow k \longrightarrow \pi (\text{Suc } k) = \pi' (\text{Suc } k') \rangle$  **and**  $\text{ret}$ :  $\langle \pi' n' = \text{return} \rangle$   
**shows**  $\langle \exists l'. \text{cs}^\pi l = \text{cs}^{\pi'} l' \rangle$  **proof cases**

**assume**  $\langle \pi l = \text{return} \rangle$   
**hence**  $\langle \text{cs}^\pi l = \text{cs}^{\pi'} n' \rangle$  **using** *ret cs-return* **by** *presburger*  
**thus**  $\langle ?\text{thesis} \rangle$  **by** *blast*

**next**

**assume**  $\text{nretl}$ :  $\langle \pi l \neq \text{return} \rangle$   
**have**  $\langle \pi l \in \text{nodes} \rangle$  **using** *path path-nodes* **by** *auto*  
**then obtain**  $\pi l n$  **where**  $\text{ipl}$ :  $\langle \text{is-path } \pi l \rangle$  **and**  $\pi l$ :  $\langle \pi l = \pi l 0 \rangle$  **and**  $\text{retn}$ :  $\langle \pi l n = \text{return} \rangle$  **and**  $\text{notl}$ :  $\langle \forall i > 0. \pi l i \neq \pi l \rangle$  **by** (*metis direct-path-return nretl*)  
**hence**  $\text{ip}$ :  $\langle \text{is-path } (\pi @^l \pi l) \rangle$  **and**  $l$ :  $\langle (\pi @^l \pi l) l = \pi l \rangle$  **and**  $\text{retl}$ :  $\langle (\pi @^l \pi l) (l + n) = \text{return} \rangle$  **and**  $\text{nl}$ :  $\langle \forall i > l. (\pi @^l \pi l) i \neq \pi l \rangle$  **using** *path-cons[OF path(1) ipl  $\pi l$ ]* **by** *auto*

**have**  $\pi 0'$ :  $\langle (\pi @^l \pi l) 0 = \pi' 0 \rangle$  **unfolding** *cs-0* **using**  $\pi l \pi 0$  **by** *auto*

**have**  $\text{csn}$ :  $\langle \text{cs}^{\pi @^l \pi l} (l+n) = \text{cs}^{\pi'} n' \rangle$  **using** *ret retl cs-return* **by** *metis*

**have**  $\text{eql}$ :  $\langle (\pi @^l \pi l) =_l \pi \rangle$  **by** (*metis path-append-eq-up-to*)

have  $cs!': \langle cs^{\pi @^l \pi l} \ l = cs^{\pi} \ l \rangle$  using  $eql \ cs\text{-path-swap} \ ip \ path(1)$  by  $metis$

have  $\langle 0 < n \rangle$  using  $nretl[unfolding \ \pi l] \ retn$  by  $(metis \ neq0\text{-conv})$   
hence  $ln: \langle l < l + n \rangle$  by  $simp$

have  $*$ :  $\langle \forall \ k \ k'. \ cs^{\pi @^l \pi l} \ k = cs^{\pi'} \ k' \wedge l \ cd^{\pi @^l \pi l} \rightarrow k \longrightarrow (\pi @^l \pi l) (Suc \ k) = \pi' (Suc \ k') \rangle$  **proof**  
*(rule,rule,rule)*

fix  $k \ k'$  assume  $*$ :  $\langle cs^{\pi @^l \pi l} \ k = cs^{\pi'} \ k' \wedge l \ cd^{\pi @^l \pi l} \rightarrow k \rangle$

hence  $kl: \langle k < l \rangle$  using  $is\text{-cdi-def}$  by  $auto$

hence  $\langle cs^{\pi} \ k = cs^{\pi'} \ k' \wedge l \ cd^{\pi} \rightarrow k \rangle$  using  $eql \ * \ cs\text{-path-swap-le}[OF \ ip \ path(1) \ eql, \ of \ \langle k \rangle] \ cdi\text{-path-swap-le}[OF \ path(1) \ - \ eql, \ of \ \langle l \rangle \ \langle k \rangle]$  by  $auto$

hence  $\langle \pi (Suc \ k) = \pi' (Suc \ k') \rangle$  using  $cd\text{-suc}$  by  $blast$

then show  $\langle (\pi @^l \pi l) (Suc \ k) = \pi' (Suc \ k') \rangle$  using  $cs\text{-path-swap-le}[OF \ ip \ path(1) \ eql, \ of \ \langle Suc \ k \rangle] \ kl$  by  $auto$

qed

obtain  $l'$  where  $\langle cs^{\pi @^l \pi l} \ l = cs^{\pi'} \ l' \rangle$  using  $converged\text{-cd-same-suc}[OF \ ip \ path(2) \ \pi 0' \ * \ ln \ csn]$  by  $blast$   
moreover

have  $\langle cs^{\pi @^l \pi l} \ l = cs^{\pi} \ l \rangle$  using  $eql$  by  $(metis \ cs\text{-path-swap} \ ip \ path(1))$

ultimately

show  $\langle ?thesis \rangle$  by  $metis$

qed

**lemma**  $converged\text{-cd-diverge-return}$ : **assumes**  $path: \langle is\text{-path} \ \pi \rangle \ \langle is\text{-path} \ \pi' \rangle$  **and**  $init: \langle \pi \ 0 = \pi' \ 0 \rangle$

**and**  $notin: \langle \neg (\exists l'. \ cs^{\pi} \ l = cs^{\pi'} \ l') \rangle$  **and**  $ret: \langle \pi' \ m' = return \rangle$

**obtains**  $k \ k'$  **where**  $\langle cs^{\pi} \ k = cs^{\pi'} \ k' \rangle \ \langle l \ cd^{\pi} \rightarrow k \rangle \ \langle \pi (Suc \ k) \neq \pi' (Suc \ k') \rangle$  using  $converged\text{-cd-same-suc-return}[OF \ path \ init \ - \ ret, \ of \ \langle l \rangle] \ notin$  by  $blast$

**lemma**  $returned\text{-missing-cd-or-loop}$ : **assumes**  $path: \langle is\text{-path} \ \pi \rangle \ \langle is\text{-path} \ \pi' \rangle$  **and**  $\pi 0: \langle \pi \ 0 = \pi' \ 0 \rangle$

**and**  $notin': \langle \neg (\exists k'. \ cs^{\pi} \ k = cs^{\pi'} \ k') \rangle$  **and**  $nret: \langle \forall n'. \ \pi' \ n' \neq return \rangle$  **and**  $ret: \langle \pi \ n = return \rangle$

**obtains**  $i \ i'$  **where**  $\langle i < k \rangle \ \langle cs^{\pi} \ i = cs^{\pi'} \ i' \rangle \ \langle \pi (Suc \ i) \neq \pi' (Suc \ i') \rangle \ \langle k \ cd^{\pi} \rightarrow i \vee (\forall j' > i'. \ j' \ cd^{\pi'} \rightarrow i') \rangle$

**proof** –

obtain  $f$  where  $icdf: \langle \forall i'. \ f (Suc \ i') \ icd^{\pi'} \rightarrow f \ i' \rangle$  **and**  $ran: \langle range \ f = \{i'. \ \forall j' > i'. \ j' \ cd^{\pi'} \rightarrow i'\} \rangle$  **and**  
 $icdf0: \langle \neg (\exists i'. \ f \ 0 \ cd^{\pi'} \rightarrow i') \rangle$  using  $path(2) \ path\text{-nret-inf-icd-seq} \ nret$  by  $blast$

show  $\langle thesis \rangle$  **proof**  $cases$

assume  $\langle \exists j. \ \neg (\exists i. \ cs^{\pi} \ i = cs^{\pi'} \ (f \ j)) \rangle$

then obtain  $j$  where  $ni\pi: \langle \neg (\exists i. \ cs^{\pi'} \ (f \ j) = cs^{\pi} \ i) \rangle$  by  $metis$

note  $converged\text{-cd-diverge-return}[OF \ path(2,1) \ \pi 0[symmetric] \ ni\pi \ ret]$  that

then obtain  $i \ k'$  where  $csk: \langle cs^{\pi} \ i = cs^{\pi'} \ k' \rangle$  **and**  $cdj: \langle f \ j \ cd^{\pi'} \rightarrow k' \rangle$  **and**  $div: \langle \pi (Suc \ i) \neq \pi' (Suc \ k') \rangle$

by  $metis$

have  $\langle k' \in range \ f \rangle$  using  $cdj$  **proof**  $(induction \ \langle j \rangle)$

case  $0$  thus  $\langle ?case \rangle$  using  $icdf0$  by  $blast$

next

case  $(Suc \ j)$

have  $icdfj: \langle f (Suc \ j) \ icd^{\pi'} \rightarrow f \ j \rangle$  using  $icdf$  by  $auto$

show  $\langle ?case \rangle$  **proof**  $cases$

assume  $\langle f (Suc \ j) \ icd^{\pi'} \rightarrow k' \rangle$

hence  $\langle k' = f \ j \rangle$  using  $icdfj$  by  $(metis \ icd\text{-uniq})$

thus  $\langle ?case \rangle$  by  $auto$

next

assume  $\langle \neg f (Suc \ j) \ icd^{\pi'} \rightarrow k' \rangle$

hence  $\langle f \ j \ cd^{\pi'} \rightarrow k' \rangle$  using  $cd\text{-impl-icd-cd}[OF \ Suc.\prems \ icdfj]$  by  $auto$

thus  $\langle ?case \rangle$  using *Suc.IH* by *auto*  
 qed  
 qed  
 hence *allddep*:  $\langle \forall i' > k'. i' \text{ cd}^{\pi'} \rightarrow k' \rangle$  using *ran* by *auto*  
 show  $\langle thesis \rangle$  proof *cases*  
 assume  $\langle i < k \rangle$  with *allddep* that[*OF - csk div*] show  $\langle thesis \rangle$  by *blast*  
 next  
 assume  $\langle \neg i < k \rangle$   
 hence *ki*:  $\langle k \leq i \rangle$  by *auto*  
 have  $\langle k \neq i \rangle$  using *notin' csk* by *auto*  
 hence *ki'*:  $\langle k < i \rangle$  using *ki* by *auto*  
 obtain *ka k'* where  $\langle \text{cs}^{\pi} ka = \text{cs}^{\pi'} k' \rangle \langle k \text{ cd}^{\pi} \rightarrow ka \rangle \langle \pi (Suc\ ka) \neq \pi' (Suc\ k') \rangle$   
 using *converged-cd-diverge[OF path  $\pi 0$  notin' ki' csk]* by *blast*  
 moreover  
 hence  $\langle ka < k \rangle$  unfolding *is-cdi-def* by *auto*  
 ultimately  
 show  $\langle ?thesis \rangle$  using *that* by *blast*  
 qed  
 next  
 assume  $\langle \neg (\exists j. \neg (\exists i. \text{cs}^{\pi} i = \text{cs}^{\pi'} (f\ j))) \rangle$   
 hence *allin*:  $\langle \forall j. (\exists i. \text{cs}^{\pi} i = \text{cs}^{\pi'} (f\ j)) \rangle$  by *blast*  
 define *f'* where *f'*:  $\langle f' \equiv \lambda j. (SOME\ i. \text{cs}^{\pi} i = \text{cs}^{\pi'} (f\ j)) \rangle$   
 have  $\langle \forall i. f' i < f' (Suc\ i) \rangle$  proof  
 fix *i*  
 have *csi*:  $\langle \text{cs}^{\pi'} (f\ i) = \text{cs}^{\pi} (f' i) \rangle$  unfolding *f'* using *allin* by (*metis (mono-tags) someI-ex*)  
 have *cssuci*:  $\langle \text{cs}^{\pi'} (f\ (Suc\ i)) = \text{cs}^{\pi} (f' (Suc\ i)) \rangle$  unfolding *f'* using *allin* by (*metis (mono-tags) someI-ex*)  
 have *fi*:  $\langle f\ i < f\ (Suc\ i) \rangle$  using *icdf unfolding is-icdi-def is-cdi-def* by *auto*  
 have  $\langle f\ (Suc\ i) \text{ cd}^{\pi'} \rightarrow f\ i \rangle$  using *icdf unfolding is-icdi-def* by *blast*  
 hence *nreti*:  $\langle \pi' (f\ i) \neq \text{return} \rangle$  by (*metis cd-not-ret*)  
 show  $\langle f' i < f' (Suc\ i) \rangle$  using *cs-order[OF path(2,1) csi cssuci nreti fi]* .  
 qed  
 hence *kle*:  $\langle k < f' (Suc\ k) \rangle$  using *mono-ge-id[of f' Suc k]* by *auto*  
 have *cssk*:  $\langle \text{cs}^{\pi} (f' (Suc\ k)) = \text{cs}^{\pi'} (f\ (Suc\ k)) \rangle$  unfolding *f'* using *allin* by (*metis (mono-tags) someI-ex*)  
 obtain *ka k'* where  $\langle \text{cs}^{\pi} ka = \text{cs}^{\pi'} k' \rangle \langle k \text{ cd}^{\pi} \rightarrow ka \rangle \langle \pi (Suc\ ka) \neq \pi' (Suc\ k') \rangle$   
 using *converged-cd-diverge[OF path  $\pi 0$  notin' kle cssk]* by *blast*  
 moreover  
 hence  $\langle ka < k \rangle$  unfolding *is-cdi-def* by *auto*  
 ultimately  
 show  $\langle ?thesis \rangle$  using *that* by *blast*  
 qed  
 qed  
 lemma *missing-cd-or-loop*: assumes *path*:  $\langle is\text{-path}\ \pi \rangle \langle is\text{-path}\ \pi' \rangle$  and  $\pi 0$ :  $\langle \pi\ 0 = \pi'\ 0 \rangle$  and *notin'*:  $\langle \neg (\exists k'. \text{cs}^{\pi} k = \text{cs}^{\pi'} k') \rangle$   
 obtains *i i'* where  $\langle i < k \rangle \langle \text{cs}^{\pi} i = \text{cs}^{\pi'} i' \rangle \langle \pi (Suc\ i) \neq \pi' (Suc\ i') \rangle \langle k \text{ cd}^{\pi} \rightarrow i \vee (\forall j' > i'. j' \text{ cd}^{\pi'} \rightarrow i') \rangle$   
 proof *cases*  
 assume  $\langle \exists n'. \pi' n' = \text{return} \rangle$   
 then obtain *n'* where *retn*:  $\langle \pi' n' = \text{return} \rangle$  by *blast*  
 note *converged-cd-diverge-return[OF path  $\pi 0$  notin' retn]*  
 then obtain *ka k'* where  $\langle \text{cs}^{\pi} ka = \text{cs}^{\pi'} k' \rangle \langle k \text{ cd}^{\pi} \rightarrow ka \rangle \langle \pi (Suc\ ka) \neq \pi' (Suc\ k') \rangle$  by *blast*  
 moreover  
 hence  $\langle ka < k \rangle$  unfolding *is-cdi-def* by *auto*  
 ultimately show  $\langle thesis \rangle$  using *that* by *simp*

next

assume  $\langle \neg (\exists n'. \pi' n' = \text{return}) \rangle$

hence *notret*:  $\langle \forall n'. \pi' n' \neq \text{return} \rangle$  by *auto*

then obtain  $\pi l n$  where *ipl*:  $\langle \text{is-path } \pi l \rangle$  and  $\pi l$ :  $\langle \pi k = \pi l 0 \rangle$  and *retn*:  $\langle \pi l n = \text{return} \rangle$  using *reaching-ret path(1) path-nodes* by *metis*

hence *ip*:  $\langle \text{is-path } (\pi @^k \pi l) \rangle$  and *l*:  $\langle (\pi @^k \pi l) k = \pi k \rangle$  and *retl*:  $\langle (\pi @^k \pi l) (k + n) = \text{return} \rangle$  using *path-cons[OF path(1) ipl  $\pi l$ ]* by *auto*

have  $\pi 0'$ :  $\langle (\pi @^k \pi l) 0 = \pi' 0 \rangle$  unfolding *cs-0* using  $\pi l \pi 0$  by *auto*

have *eql*:  $\langle (\pi @^k \pi l) =_k \pi \rangle$  by (*metis path-append-eq-up-to*)

have *csl'*:  $\langle cs^{\pi @^k \pi l} k = cs^\pi k \rangle$  using *eql cs-path-swap ip path(1)* by *metis*

hence *notin*:  $\langle \neg (\exists k'. cs^{\pi @^k \pi l} k = cs^{\pi'} k') \rangle$  using *notin'* by *auto*

obtain *i i'* where *\**:  $\langle i < k \rangle$  and *csi*:  $\langle cs^{\pi @^k \pi l} i = cs^{\pi'} i' \rangle$  and *suci*:  $\langle (\pi @^k \pi l) (\text{Suc } i) \neq \pi' (\text{Suc } i') \rangle$

and *cdloop*:  $\langle k \text{ cd}^{\pi @^k \pi l} \rightarrow i \vee (\forall j' > i'. j' \text{ cd}^{\pi'} \rightarrow i') \rangle$

using *returned-missing-cd-or-loop[OF ip path(2)  $\pi 0'$  notin notret retl]* by *blast*

have  $\langle i \neq k \rangle$  using *notin csi* by *auto*

hence *ik*:  $\langle i < k \rangle$  using *\** by *auto*

hence  $\langle cs^\pi i = cs^{\pi'} i' \rangle$  using *csi cs-path-swap-le[OF ip path(1) eql]* by *auto*

moreover

have  $\langle \pi (\text{Suc } i) \neq \pi' (\text{Suc } i') \rangle$  using *ik eq-up-to-apply[OF eql, of  $\langle \text{Suc } i \rangle$  suci]* by *auto*

moreover

have  $\langle k \text{ cd}^\pi \rightarrow i \vee (\forall j' > i'. j' \text{ cd}^{\pi'} \rightarrow i') \rangle$  using *cdloop cdi-path-swap-le[OF path(1) - eql, of  $\langle k \rangle \langle i \rangle$ ]* by *auto*

ultimately

show *thesis* using *that[OF \*]* by *blast*

qed

lemma *path-shift-set-cd*: assumes  $\langle \text{is-path } \pi \rangle$  shows  $\langle \{k + j \mid j . n \text{ cd}^{\pi @^k} \rightarrow j\} = \{i. (k+n) \text{ cd}^\pi \rightarrow i \wedge k \leq i\} \rangle$

proof –

{ fix *i*

assume  $\langle i \in \{k+j \mid j . n \text{ cd}^{\pi @^k} \rightarrow j\} \rangle$

then obtain *j* where  $\langle i = k+j \rangle \langle n \text{ cd}^{\pi @^k} \rightarrow j \rangle$  by *auto*

hence  $\langle k+n \text{ cd}^\pi \rightarrow i \wedge k \leq i \rangle$  using *cd-path-shift[OF - assms, of  $\langle k \rangle \langle k+j \rangle \langle k+n \rangle$ ]* by *simp*

hence  $\langle i \in \{i. k+n \text{ cd}^\pi \rightarrow i \wedge k \leq i\} \rangle$  by *blast*

}

moreover

{ fix *i*

assume  $\langle i \in \{i. k+n \text{ cd}^\pi \rightarrow i \wedge k \leq i\} \rangle$

hence *\**:  $\langle k+n \text{ cd}^\pi \rightarrow i \wedge k \leq i \rangle$  by *blast*

then obtain *j* where *i*:  $\langle i = k+j \rangle$  by (*metis le-Suc-ex*)

hence  $\langle k+n \text{ cd}^\pi \rightarrow k+j \rangle$  using *\** by *auto*

hence  $\langle n \text{ cd}^{\pi @^k} \rightarrow j \rangle$  using *cd-path-shift[OF - assms, of  $\langle k \rangle \langle k+j \rangle \langle k+n \rangle$ ]* by *simp*

hence  $\langle i \in \{k+j \mid j . n \text{ cd}^{\pi @^k} \rightarrow j\} \rangle$  using *i* by *simp*

}

ultimately show *thesis* by *blast*

qed

lemma *cs-path-shift-set-cd*: assumes *path*:  $\langle \text{is-path } \pi \rangle$  shows  $\langle cs^{\pi @^k} n = \text{map } \pi (\text{sorted-list-of-set } \{i. k+n\}) \rangle$



$cd^\pi \rightarrow i \wedge k \leq i$ ) @  $[\pi (k+n)]$

**proof** –

**have** *mono*:  $\langle \forall n m. n < m \rightarrow k + n < k + m \rangle$  **by** *auto*

**have** *fin*:  $\langle \text{finite } \{i. n \text{ cd}^\pi \ll k \rightarrow i\} \rangle$  **unfolding** *is-cdi-def* **by** *auto*

**have** *\**:  $\langle (\lambda x. k+x) \{i. n \text{ cd}^\pi \ll k \rightarrow i\} = \{k+i \mid i. n \text{ cd}^\pi \ll k \rightarrow i\} \rangle$  **by** *auto*

**have**  $\langle cs^{\pi \ll k} n = \text{map } (\pi \ll k) (\text{sorted-list-of-set } \{i. n \text{ cd}^\pi \ll k \rightarrow i\}) @ [(\pi \ll k) n] \rangle$  **using** *cs-sorted-list-of-cd'* **by** *blast*

**also**

**have**  $\langle \dots = \text{map } \pi (\text{map } (\lambda x. k+x) (\text{sorted-list-of-set } \{i. n \text{ cd}^\pi \ll k \rightarrow i\})) @ [\pi (k+n)] \rangle$  **by** *auto*

**also**

**have**  $\langle \dots = \text{map } \pi (\text{sorted-list-of-set } ((\lambda x. k+x) \{i. n \text{ cd}^\pi \ll k \rightarrow i\})) @ [\pi (k+n)] \rangle$  **using** *sorted-list-of-set-map-mono*[*OF mono fin*] **by** *auto*

**also**

**have**  $\langle \dots = \text{map } \pi (\text{sorted-list-of-set } (\{k+i \mid i. n \text{ cd}^\pi \ll k \rightarrow i\})) @ [\pi (k+n)] \rangle$  **using** *\** **by** *auto*

**also**

**have**  $\langle \dots = \text{map } \pi (\text{sorted-list-of-set } (\{i. k+n \text{ cd}^\pi \rightarrow i \wedge k \leq i\})) @ [\pi (k+n)] \rangle$  **using** *path-shift-set-cd*[*OF path*] **by** *auto*

**finally**

**show**  $\langle ?thesis \rangle$  .

**qed**

**lemma** *cs-split-shift-cd*: **assumes**  $\langle n \text{ cd}^\pi \rightarrow j \rangle$  **and**  $\langle j < k \rangle$  **and**  $\langle k < n \rangle$  **and**  $\langle \forall j' < k. n \text{ cd}^\pi \rightarrow j' \rightarrow j' \leq j \rangle$  **shows**  $\langle cs^\pi n = cs^\pi j @ cs^{\pi \ll k} (n-k) \rangle$

**proof** –

**have** *path*:  $\langle \text{is-path } \pi \rangle$  **using** *assms* **unfolding** *is-cdi-def* **by** *auto*

**have** *1*:  $\langle \{i. n \text{ cd}^\pi \rightarrow i\} = \{i. n \text{ cd}^\pi \rightarrow i \wedge i < k\} \cup \{i. n \text{ cd}^\pi \rightarrow i \wedge k \leq i\} \rangle$  **by** *auto*

**have** *le*:  $\langle \forall i \in \{i. n \text{ cd}^\pi \rightarrow i \wedge i < k\}. \forall j \in \{i. n \text{ cd}^\pi \rightarrow i \wedge k \leq i\}. i < j \rangle$  **by** *auto*

**have** *2*:  $\langle \{i. n \text{ cd}^\pi \rightarrow i \wedge i < k\} = \{i. j \text{ cd}^\pi \rightarrow i\} \cup \{j\} \rangle$  **proof** –

{ **fix** *i* **assume**  $\langle i \in \{i. n \text{ cd}^\pi \rightarrow i \wedge i < k\} \rangle$

**hence** *cd*:  $\langle n \text{ cd}^\pi \rightarrow i \rangle$  **and** *ik*:  $\langle i < k \rangle$  **by** *auto*

**have**  $\langle i \in \{i. j \text{ cd}^\pi \rightarrow i\} \cup \{j\} \rangle$  **proof** *cases*

**assume**  $\langle i < j \rangle$  **hence**  $\langle j \text{ cd}^\pi \rightarrow i \rangle$  **by** (*metis is-cdi-def assms(1) cd cdi-prefix nat-less-le*)

**thus**  $\langle ?thesis \rangle$  **by** *simp*

**next**

**assume**  $\langle \neg i < j \rangle$

**moreover**

**have**  $\langle i \leq j \rangle$  **using** *assms(4) ik cd* **by** *auto*

**ultimately**

**show**  $\langle ?thesis \rangle$  **by** *auto*

**qed**

}

**moreover**

{ **fix** *i* **assume**  $\langle i \in \{i. j \text{ cd}^\pi \rightarrow i\} \cup \{j\} \rangle$

**hence**  $\langle j \text{ cd}^\pi \rightarrow i \vee i = j \rangle$  **by** *auto*

**hence**  $\langle i \in \{i. n \text{ cd}^\pi \rightarrow i \wedge i < k\} \rangle$  **using** *assms(1,2) cd-trans*[*OF - assms(1)*] **apply** *auto* **unfolding**

*is-cdi-def*

**by** (*metis (poly-guards-query) diff-diff-cancel diff-is-0-eq le-refl le-trans nat-less-le*)

}

**ultimately** **show**  $\langle ?thesis \rangle$  **by** *blast*

**qed**

**have**  $\langle cs^\pi n = \text{map } \pi (\text{sorted-list-of-set } \{i. n \text{ cd}^\pi \rightarrow i\}) @ [\pi n] \rangle$  **using** *cs-sorted-list-of-cd'* **by** *simp*

**also**

**have**  $\langle \dots = \text{map } \pi (\text{sorted-list-of-set } (\{i. n \text{ cd}^\pi \rightarrow i \wedge i < k\} \cup \{i. n \text{ cd}^\pi \rightarrow i \wedge k \leq i\})) @ [\pi n] \rangle$  **using** *1* **by** *metis*

**also**  
**have**  $\langle \dots = \text{map } \pi ((\text{sorted-list-of-set } \{i. n \text{ cd}^\pi \rightarrow i \wedge i < k\}) @ (\text{sorted-list-of-set } \{i. n \text{ cd}^\pi \rightarrow i \wedge k \leq i\}))$   
 $@ [\pi n] \rangle$   
**using** *sorted-list-of-set-append*[*OF - - le*] *is-cdi-def* **by** *auto*  
**also**  
**have**  $\langle \dots = (\text{map } \pi (\text{sorted-list-of-set } \{i. n \text{ cd}^\pi \rightarrow i \wedge i < k\})) @ (\text{map } \pi (\text{sorted-list-of-set } \{i. n \text{ cd}^\pi \rightarrow i \wedge$   
 $k \leq i\})) @ [\pi n] \rangle$  **by** *auto*  
**also**  
**have**  $\langle \dots = \text{cs}^\pi j @ (\text{map } \pi (\text{sorted-list-of-set } \{i. n \text{ cd}^\pi \rightarrow i \wedge k \leq i\})) @ [\pi n] \rangle$  **unfolding** 2 **using**  
*cs-sorted-list-of-cd* **by** *auto*  
**also**  
**have**  $\langle \dots = \text{cs}^\pi j @ \text{cs}^{\pi \ll k} (n-k) \rangle$  **using** *cs-path-shift-set-cd*[*OF path, of*  $\langle k \rangle \langle n-k \rangle$ ] *assms*(3) **by** *auto*  
**finally**  
**show**  $\langle ?thesis \rangle$  .  
**qed**

**lemma** *cs-split-shift-nocd*: **assumes**  $\langle \text{is-path } \pi \rangle$  **and**  $\langle k < n \rangle$  **and**  $\langle \forall j. n \text{ cd}^\pi \rightarrow j \longrightarrow k \leq j \rangle$  **shows**  $\langle \text{cs}^\pi n =$   
 $\text{cs}^{\pi \ll k} (n-k) \rangle$

**proof** –

**have** *path*:  $\langle \text{is-path } \pi \rangle$  **using** *assms* **by** *auto*  
**have** 1:  $\langle \{i. n \text{ cd}^\pi \rightarrow i\} = \{i. n \text{ cd}^\pi \rightarrow i \wedge i < k\} \cup \{i. n \text{ cd}^\pi \rightarrow i \wedge k \leq i\} \rangle$  **by** *auto*  
**have** *le*:  $\langle \forall i \in \{i. n \text{ cd}^\pi \rightarrow i \wedge i < k\}. \forall j \in \{i. n \text{ cd}^\pi \rightarrow i \wedge k \leq i\}. i < j \rangle$  **by** *auto*  
**have** 2:  $\langle \{i. n \text{ cd}^\pi \rightarrow i \wedge i < k\} = \{\} \rangle$  **using** *assms* **by** *auto*  
  
**have**  $\langle \text{cs}^\pi n = \text{map } \pi (\text{sorted-list-of-set } \{i. n \text{ cd}^\pi \rightarrow i\}) @ [\pi n] \rangle$  **using** *cs-sorted-list-of-cd'* **by** *simp*  
**also**  
**have**  $\langle \dots = \text{map } \pi (\text{sorted-list-of-set } (\{i. n \text{ cd}^\pi \rightarrow i \wedge i < k\} \cup \{i. n \text{ cd}^\pi \rightarrow i \wedge k \leq i\})) @ [\pi n] \rangle$  **using** 1  
**by** *metis*  
**also**  
**have**  $\langle \dots = \text{map } \pi (\text{sorted-list-of-set } \{i. n \text{ cd}^\pi \rightarrow i \wedge k \leq i\}) @ [\pi n] \rangle$   
**unfolding** 2 **by** *auto*  
**also**  
**have**  $\langle \dots = \text{cs}^{\pi \ll k} (n-k) \rangle$  **using** *cs-path-shift-set-cd*[*OF path, of*  $\langle k \rangle \langle n-k \rangle$ ] *assms*(2) **by** *auto*  
**finally** **show**  $\langle ?thesis \rangle$  .  
**qed**

**lemma** *shifted-cs-eq-is-eq*: **assumes**  $\langle \text{is-path } \pi \rangle$  **and**  $\langle \text{is-path } \pi' \rangle$  **and**  $\langle \text{cs}^\pi k = \text{cs}^{\pi'} k' \rangle$  **and**  $\langle \text{cs}^{\pi \ll k} n = \text{cs}^{\pi' \ll k'} n' \rangle$   
 $n' \rangle$  **shows**  $\langle \text{cs}^\pi (k+n) = \text{cs}^{\pi'} (k'+n') \rangle$

**proof** (*rule ccontr*)

**note** *path* = *assms*(1,2)

**note** *csk* = *assms*(3)

**note** *csn* = *assms*(4)

**assume** *ne*:  $\langle \text{cs}^\pi (k+n) \neq \text{cs}^{\pi'} (k'+n') \rangle$

**have** *nretkn*:  $\langle \pi (k+n) \neq \text{return} \rangle$  **proof**

**assume** 1:  $\langle \pi (k+n) = \text{return} \rangle$

**hence**  $\langle (\pi \ll k) n = \text{return} \rangle$  **by** *auto*

**hence**  $\langle (\pi' \ll k') n' = \text{return} \rangle$  **using** *last-cs* *assms*(4) **by** *metis*

**hence**  $\langle \pi' (k' + n') = \text{return} \rangle$  **by** *auto*

**thus**  $\langle \text{False} \rangle$  **using** *ne* 1 *cs-return* **by** *auto*

**qed**

**hence** *nretk*:  $\langle \pi k \neq \text{return} \rangle$  **using** *term-path-stable*[*OF* *assms*(1), *of*  $\langle k \rangle \langle k+n \rangle$ ] **by** *auto*

**have** *nretkn'*:  $\langle \pi' (k'+n') \neq \text{return} \rangle$  **proof**

**assume** 1:  $\langle \pi' (k'+n') = \text{return} \rangle$

**hence**  $\langle (\pi' \ll k') n' = \text{return} \rangle$  **by** *auto*

**hence**  $\langle (\pi \ll k) n = \text{return} \rangle$  **using** *last-cs* *assms*(4) **by** *metis*

hence  $\langle \pi (k + n) = \text{return} \rangle$  **by auto**  
 thus  $\langle \text{False} \rangle$  **using ne 1 cs-return by auto**  
**qed**  
 hence  $\text{nretk}' : \langle \pi' k' \neq \text{return} \rangle$  **using term-path-stable**[*OF assms(2)*, of  $\langle k' \rangle \langle k' + n' \rangle$ ] **by auto**  
 have  $n0 : \langle n > 0 \rangle$  **proof (rule ccontr)**  
 assume \*:  $\langle \neg 0 < n \rangle$   
 hence  $1 : \langle \text{cs}^{\pi \ll k} 0 = \text{cs}^{\pi' \ll k'} n' \rangle$  **using assms(3,4) by auto**  
 have  $\langle (\pi \ll k) 0 = (\pi' \ll k') 0 \rangle$  **using assms(3) last-cs path-shift-def by (metis monoid-add-class.add.right-neutral)**  
 hence  $\langle \text{cs}^{\pi' \ll k'} 0 = \text{cs}^{\pi \ll k} n' \rangle$  **using 1 cs-0 by metis**  
 hence  $n0' : \langle n' = 0 \rangle$  **using cs-inj**[of  $\langle \pi' \ll k' \rangle \langle 0 \rangle \langle n' \rangle$ ] \* *assms(2)* **by (metis path-shift-def assms(4) last-cs nretkn path-path-shift)**  
 thus  $\langle \text{False} \rangle$  **using ne \* assms(3) by fastforce**  
**qed**  
 have  $n0' : \langle n' > 0 \rangle$  **proof (rule ccontr)**  
 assume \*:  $\langle \neg 0 < n' \rangle$   
 hence  $1 : \langle \text{cs}^{\pi' \ll k'} 0 = \text{cs}^{\pi \ll k} n \rangle$  **using assms(3,4) by auto**  
 have  $\langle (\pi' \ll k') 0 = (\pi \ll k) 0 \rangle$  **using assms(3) last-cs path-shift-def by (metis monoid-add-class.add.right-neutral)**  
 hence  $\langle \text{cs}^{\pi \ll k} 0 = \text{cs}^{\pi' \ll k'} n \rangle$  **using 1 cs-0 by metis**  
 hence  $n0 : \langle n = 0 \rangle$  **using cs-inj**[of  $\langle \pi \ll k \rangle \langle 0 \rangle \langle n \rangle$ ] \* *assms(1)* **by (metis path-shift-def assms(4) last-cs nretkn path-path-shift)**  
 thus  $\langle \text{False} \rangle$  **using ne \* assms(3) by fastforce**  
**qed**  
 have  $\text{cdleswap}' : \langle \forall j' < k'. (k' + n') \text{cd}^{\pi'} \rightarrow j' \longrightarrow (\exists j < k. (k + n) \text{cd}^{\pi} \rightarrow j \wedge \text{cs}^{\pi} j = \text{cs}^{\pi'} j') \rangle$  **proof (rule, rule, rule, rule ccontr)**  
 fix  $j'$  **assume  $\text{jk}' : \langle j' < k' \rangle$  and  $\text{ncdj}' : \langle (k' + n') \text{cd}^{\pi'} \rightarrow j' \rangle$  and  $\text{ne} : \langle \neg (\exists j < k. k + n \text{cd}^{\pi} \rightarrow j \wedge \text{cs}^{\pi} j = \text{cs}^{\pi'} j') \rangle$**   
 hence  $\text{kcdj}' : \langle k' \text{cd}^{\pi'} \rightarrow j' \rangle$  **using cr-wn' by blast**  
  
 then obtain  $j$  where  $\text{kcdj} : \langle k \text{cd}^{\pi} \rightarrow j \rangle$  **and  $\text{csj} : \langle \text{cs}^{\pi} j = \text{cs}^{\pi'} j' \rangle$  using *csk cs-path-swap-cd path* by *metis***  
 hence  $\text{jk} : \langle j < k \rangle$  **unfolding is-cdi-def by auto**  
 have  $\text{ncdn} : \langle \neg (k + n) \text{cd}^{\pi} \rightarrow j \rangle$  **using ne csj jk by blast**  
  
 obtain  $l'$  where  $\text{lnocd}' : \langle l' = n' \vee n' \text{cd}^{\pi' \ll k'} \rightarrow l' \rangle$  **and  $\text{cslsing}' : \langle \text{cs}^{\pi' \ll k'} l' = [(\pi' \ll k') l'] \rangle$**   
**proof cases**  
 assume  $\langle \text{cs}^{\pi' \ll k'} n' = [(\pi' \ll k') n'] \rangle$  **thus  $\langle \text{thesis} \rangle$  using that**[of  $\langle n' \rangle$ ] **by auto**  
**next**  
 assume \*:  $\langle \text{cs}^{\pi' \ll k'} n' \neq [(\pi' \ll k') n'] \rangle$   
 then obtain  $x$   $ys$  where  $\langle \text{cs}^{\pi' \ll k'} n' = [x] @ ys @ [(\pi' \ll k') n'] \rangle$  **by (metis append-Cons append-Nil cs-length-g-one cs-length-one(1) neq-Nil-conv)**  
 then obtain  $l'$  where  $\langle \text{cs}^{\pi' \ll k'} l' = [x] \rangle$  **and  $\text{cdl}' : \langle n' \text{cd}^{\pi' \ll k'} \rightarrow l' \rangle$  using *cs-split***[of  $\langle \pi' \ll k' \rangle \langle n' \rangle \langle \text{Nil} \rangle \langle x \rangle \langle ys \rangle$ ] **by auto**  
 hence  $\langle \text{cs}^{\pi' \ll k'} l' = [(\pi' \ll k') l'] \rangle$  **using last-cs by (metis last.simps)**  
 thus  $\langle \text{thesis} \rangle$  **using that  $\text{cdl}'$  by auto**  
**qed**  
 hence  $\text{ln}' : \langle l' \leq n' \rangle$  **unfolding is-cdi-def by auto**  
 hence  $\text{lcdj}' : \langle k' + l' \text{cd}^{\pi'} \rightarrow j' \rangle$  **using  $\text{jk}' \text{ncdj}'$  by (metis add-le-cancel-left cdi-prefix trans-less-add1)**  
  
 obtain  $l$  where  $\text{lnocd} : \langle l = n \vee n \text{cd}^{\pi \ll k} \rightarrow l \rangle$  **and  $\text{csl} : \langle \text{cs}^{\pi \ll k} l = \text{cs}^{\pi' \ll k'} l' \rangle$  using *lnocd'* **proof****  
 assume  $\langle l' = n' \rangle$  **thus  $\langle \text{thesis} \rangle$  using *csn that***[of  $\langle n' \rangle$ ] **by auto**  
**next**  
 assume  $\langle n' \text{cd}^{\pi' \ll k'} \rightarrow l' \rangle$   
 then obtain  $l$  where  $\langle n \text{cd}^{\pi \ll k} \rightarrow l \rangle$   $\langle \text{cs}^{\pi \ll k} l = \text{cs}^{\pi' \ll k'} l' \rangle$  **using *cs-path-swap-cd path csn* by (metis**

*path-path-shift*)

**thus**  $\langle thesis \rangle$  **using that by auto**  
**qed**

**have** *cslsing*:  $\langle cs^{\pi \ll k} l = [(\pi \ll k) l] \rangle$  **using** *cslsing'* *last-cs csl last.simps* **by metis**

**have** *ln*:  $\langle l \leq n \rangle$  **using** *lnocd unfolding is-cdi-def* **by auto**

**hence** *nretkl*:  $\langle \pi (k + l) \neq return \rangle$  **using** *term-path-stable*[of  $\langle \pi \rangle \langle k+l \rangle \langle k+n \rangle$ ] *nretkn path(1)* **by auto**

**have** \*:  $\langle n \text{ cd}^{\pi \ll k} \rightarrow l \implies k+n \text{ cd}^{\pi} \rightarrow k+l \rangle$  **using** *cd-path-shift*[of  $\langle k \rangle \langle k+l \rangle \langle \pi \rangle \langle k+n \rangle$ ] *path(1)* **by auto**

**have** *ncdl*:  $\langle \neg (k+l) \text{ cd}^{\pi} \rightarrow j \rangle$  **apply rule using** *lnocd apply rule using* *ncdn apply blast using* *cd-trans ncdn \** **by blast**

**hence**  $\langle \exists i \in \{j..k+l\}. \pi i = ipd (\pi j) \rangle$  **unfolding** *is-cdi-def using path(1) jk nretkl* **by auto**

**hence**  $\langle \exists i \in \{k<..k+l\}. \pi i = ipd (\pi j) \rangle$  **using** *kcdj unfolding is-cdi-def* **by force**

**then obtain** *i* **where** *ki*:  $\langle k < i \rangle$  **and** *il*:  $\langle i \leq k+l \rangle$  **and** *ipdi*:  $\langle \pi i = ipd (\pi j) \rangle$  **by force**

**hence**  $\langle (\pi \ll k) (i-k) = ipd (\pi j) \rangle \langle i-k \leq l \rangle$  **by auto**

**hence** *pd*:  $\langle (\pi \ll k) l \text{ pd} \rightarrow ipd (\pi j) \rangle$  **using** *cs-single-pd-intermed*[*OF - cslsing*] *path(1) path-path-shift* **by metis**

**moreover**

**have**  $\langle (\pi \ll k) l = \pi' (k' + l') \rangle$  **using** *csl last-cs* **by** (*metis path-shift-def*)

**moreover**

**have**  $\langle \pi j = \pi' j' \rangle$  **using** *csj last-cs* **by metis**

**ultimately**

**have**  $\langle \pi' (k'+l') \text{ pd} \rightarrow ipd (\pi' j') \rangle$  **by simp**

**moreover**

**have**  $\langle ipd (\pi' j') \text{ pd} \rightarrow \pi' (k'+l') \rangle$  **using** *ipd-pd-cd*[*OF lcdj'*].

**ultimately**

**have**  $\langle \pi' (k'+l') = ipd (\pi' j') \rangle$  **using** *pd-antisym* **by auto**

**thus**  $\langle False \rangle$  **using** *lcdj'* **unfolding** *is-cdi-def* **by force**

**qed**

— Symmetric version of the above statement

**have** *cdleswap*:  $\langle \forall j < k. (k+n) \text{ cd}^{\pi} \rightarrow j \implies (\exists j' < k'. (k'+n') \text{ cd}^{\pi'} \rightarrow j' \wedge cs^{\pi} j = cs^{\pi'} j') \rangle$  **proof** (*rule,rule,rule, rule ccontr*)

**fix** *j* **assume** *jk*:  $\langle j < k \rangle$  **and** *ncdj*:  $\langle (k+n) \text{ cd}^{\pi} \rightarrow j \rangle$  **and** *ne*:  $\langle \neg (\exists j' < k'. k' + n' \text{ cd}^{\pi'} \rightarrow j' \wedge cs^{\pi} j = cs^{\pi'} j') \rangle$

**hence** *kcdj*:  $\langle k \text{ cd}^{\pi} \rightarrow j \rangle$  **using** *cr-wn'* **by blast**

**then obtain** *j'* **where** *kcdj'*:  $\langle k' \text{ cd}^{\pi'} \rightarrow j' \rangle$  **and** *csj*:  $\langle cs^{\pi} j = cs^{\pi'} j' \rangle$  **using** *csk cs-path-swap-cd path* **by metis**

**hence** *jk'*:  $\langle j' < k' \rangle$  **unfolding** *is-cdi-def* **by auto**

**have** *ncdn'*:  $\langle \neg (k'+n') \text{ cd}^{\pi'} \rightarrow j' \rangle$  **using** *ne csj jk'* **by blast**

**obtain** *l* **where** *lnocd*:  $\langle l = n \vee n \text{ cd}^{\pi \ll k} \rightarrow l \rangle$  **and** *cslsing*:  $\langle cs^{\pi \ll k} l = [(\pi \ll k) l] \rangle$

**proof cases**

**assume**  $\langle cs^{\pi \ll k} n = [(\pi \ll k) n] \rangle$  **thus**  $\langle thesis \rangle$  **using that**[of  $\langle n \rangle$ ] **by auto**

**next**

**assume** \*:  $\langle cs^{\pi \ll k} n \neq [(\pi \ll k) n] \rangle$

**then obtain** *x ys* **where**  $\langle cs^{\pi \ll k} n = [x]@ys@[ (\pi \ll k) n] \rangle$  **by** (*metis append-Cons append-Nil cs-length-g-one cs-length-one(1) neq-Nil-conv*)

then obtain  $l$  where  $\langle cs^{\pi \ll k} l = [x] \rangle$  and  $cdl: \langle n \text{ cd}^{\pi \ll k} \rightarrow l \rangle$  using  $cs\text{-split}[of \langle \pi \ll k \rangle \langle n \rangle \langle Nil \rangle \langle x \rangle \langle ys \rangle]$  by *auto*

hence  $\langle cs^{\pi \ll k} l = [(\pi \ll k) l] \rangle$  using *last-cs* by (*metis last.simps*)  
 thus  $\langle thesis \rangle$  using *that cdl* by *auto*

qed

hence  $ln: \langle l \leq n \rangle$  unfolding *is-cdi-def* by *auto*

hence  $lcdj: \langle k+l \text{ cd}^{\pi} \rightarrow j \rangle$  using  $jk \text{ ncdj}$  by (*metis add-le-cancel-left cdi-prefix trans-less-add1*)

obtain  $l'$  where  $lnocd': \langle l' = n' \vee n' \text{ cd}^{\pi' \ll k'} \rightarrow l' \rangle$  and  $csl: \langle cs^{\pi \ll k} l = cs^{\pi' \ll k'} l' \rangle$  using *lnocd proof*  
 assume  $\langle l = n \rangle$  thus  $\langle thesis \rangle$  using *csn that[of \langle n' \rangle]* by *auto*

next

assume  $\langle n \text{ cd}^{\pi \ll k} \rightarrow l \rangle$

then obtain  $l'$  where  $\langle n' \text{ cd}^{\pi' \ll k'} \rightarrow l' \rangle$   $\langle cs^{\pi \ll k} l = cs^{\pi' \ll k'} l' \rangle$  using *cs-path-swap-cd path csn* by (*metis path-path-shift*)

thus  $\langle thesis \rangle$  using *that* by *auto*

qed

have  $cslsing': \langle cs^{\pi' \ll k'} l' = [(\pi' \ll k') l'] \rangle$  using *cslsing last-cs csl last.simps* by *metis*

have  $ln': \langle l' \leq n' \rangle$  using *lnocd' unfolding is-cdi-def* by *auto*

hence  $nretkl': \langle \pi' (k' + l') \neq \text{return} \rangle$  using *term-path-stable[of \langle \pi' \rangle \langle k'+l' \rangle \langle k'+n' \rangle]* *nretkn' path(2)* by *auto*

have  $*$ :  $\langle n' \text{ cd}^{\pi' \ll k'} \rightarrow l' \implies k'+n' \text{ cd}^{\pi'} \rightarrow k'+l' \rangle$  using *cd-path-shift[of \langle k' \rangle \langle k'+l' \rangle \langle \pi' \rangle \langle k'+n' \rangle]* *path(2)* by *auto*

have  $ncdl': \langle \neg (k'+l') \text{ cd}^{\pi'} \rightarrow j' \rangle$  apply *rule* using *lnocd' apply rule* using *ncdn' apply blast* using *cd-trans ncdn' \** by *blast*

hence  $\langle \exists i' \in \{j'..k'+l'\}. \pi' i' = ipd (\pi' j') \rangle$  unfolding *is-cdi-def* using *path(2) jk' nretkl'* by *auto*

hence  $\langle \exists i' \in \{k'..k'+l'\}. \pi' i' = ipd (\pi' j') \rangle$  using *kcdj' unfolding is-cdi-def* by *force*

then obtain  $i'$  where  $ki': \langle k' < i' \rangle$  and  $il': \langle i' \leq k'+l' \rangle$  and  $ipdi: \langle \pi' i' = ipd (\pi' j') \rangle$  by *force*

hence  $\langle (\pi' \ll k') (i' - k') = ipd (\pi' j') \rangle$   $\langle i' - k' \leq l' \rangle$  by *auto*

hence  $pd: \langle (\pi' \ll k') l' \text{ pd} \rightarrow ipd (\pi' j') \rangle$  using *cs-single-pd-intermed[OF - cslsing'] path(2) path-path-shift* by *metis*

moreover

have  $\langle (\pi' \ll k') l' = \pi (k + l) \rangle$  using *csl last-cs* by (*metis path-shift-def*)

moreover

have  $\langle \pi' j' = \pi j \rangle$  using *csj last-cs* by *metis*

ultimately

have  $\langle \pi (k+l) \text{ pd} \rightarrow ipd (\pi j) \rangle$  by *simp*

moreover

have  $\langle ipd (\pi j) \text{ pd} \rightarrow \pi (k+l) \rangle$  using *ipd-pd-cd[OF lcdj]* .

ultimately

have  $\langle \pi (k+l) = ipd (\pi j) \rangle$  using *pd-antisym* by *auto*

thus  $\langle False \rangle$  using *lcdj unfolding is-cdi-def* by *force*

qed

have  $cdle: \langle \exists j. (k+n) \text{ cd}^{\pi} \rightarrow j \wedge j < k \rangle$  (is  $\langle \exists j. ?P j \rangle$ ) **proof** (*rule ccontr*)

assume  $\langle \neg (\exists j. (k+n) \text{ cd}^{\pi} \rightarrow j \wedge j < k) \rangle$

hence  $allge: \langle \forall j. (k+n) \text{ cd}^{\pi} \rightarrow j \implies k \leq j \rangle$  by *auto*

have  $allge': \langle \forall j'. (k'+n') \text{ cd}^{\pi'} \rightarrow j' \implies k' \leq j' \rangle$  **proof** (*rule, rule, rule ccontr*)

fix  $j'$

**assume**  $*$ :  $\langle k' + n' \text{ cd}^{\pi'} \rightarrow j' \rangle$  **and**  $\langle \neg k' \leq j' \rangle$   
**then obtain**  $j$  **where**  $\langle j < k \rangle \langle (k+n) \text{ cd}^{\pi} \rightarrow j \rangle$  **using**  $\text{cdleswap}'$  **by** (*metis le-neq-implies-less nat-le-linear*)  
**thus**  $\langle \text{False} \rangle$  **using**  $\text{allge}$  **by**  $\text{auto}$   
**qed**  
**have**  $\langle \text{cs}^{\pi} (k + n) = \text{cs}^{\pi} \ll k \ n \rangle$  **using**  $\text{cs-split-shift-nocd}[OF \text{ assms}(1) - \text{allge}] \ n0$  **by**  $\text{auto}$   
**moreover**  
**have**  $\langle \text{cs}^{\pi'} (k' + n') = \text{cs}^{\pi'} \ll k' \ n' \rangle$  **using**  $\text{cs-split-shift-nocd}[OF \text{ assms}(2) - \text{allge}] \ n0'$  **by**  $\text{auto}$   
**ultimately**  
**show**  $\langle \text{False} \rangle$  **using**  $\text{ne assms}(4)$  **by**  $\text{auto}$   
**qed**

**define**  $j$  **where**  $\langle j == \text{GREATEST } j. (k+n) \text{ cd}^{\pi} \rightarrow j \wedge j < k \rangle$   
**have**  $\text{cdj}: \langle (k+n) \text{ cd}^{\pi} \rightarrow j \rangle$  **and**  $\text{jk}: \langle j < k \rangle$  **and**  $\text{jge}: \langle \forall j' < k. (k+n) \text{ cd}^{\pi} \rightarrow j' \longrightarrow j' \leq j \rangle$  **proof** –  
**have**  $\text{bound}: \langle \forall y. ?P y \longrightarrow y \leq k \rangle$  **by**  $\text{auto}$   
**show**  $\langle (k+n) \text{ cd}^{\pi} \rightarrow j \rangle$  **using**  $\text{GreatestI-nat}[of \langle ?P \rangle] \ j\text{-def bound cdle}$  **by**  $\text{blast}$   
**show**  $\langle j < k \rangle$  **using**  $\text{GreatestI-nat}[of \langle ?P \rangle] \ \text{bound } j\text{-def cdle}$  **by**  $\text{blast}$   
**show**  $\langle \forall j' < k. (k+n) \text{ cd}^{\pi} \rightarrow j' \longrightarrow j' \leq j \rangle$  **using**  $\text{Greatest-le-nat}[of \langle ?P \rangle] \ \text{bound } j\text{-def}$  **by**  $\text{blast}$   
**qed**

**obtain**  $j'$  **where**  $\text{cdj}': \langle (k'+n') \text{ cd}^{\pi'} \rightarrow j' \rangle$  **and**  $\text{csj}: \langle \text{cs}^{\pi} j = \text{cs}^{\pi'} j' \rangle$  **and**  $\text{jk}': \langle j' < k' \rangle$  **using**  $\text{cdleswap}$   $\text{cdj } \text{jk}$   
**by**  $\text{blast}$   
**have**  $\text{jge}': \langle \forall i' < k'. (k'+n') \text{ cd}^{\pi'} \rightarrow i' \longrightarrow i' \leq j' \rangle$  **proof**( $\text{rule}, \text{rule}, \text{rule}$ )  
**fix**  $i'$   
**assume**  $\text{ik}': \langle i' < k' \rangle$  **and**  $\text{cdi}': \langle k' + n' \text{ cd}^{\pi'} \rightarrow i' \rangle$   
**then obtain**  $i$  **where**  $\text{cdi}: \langle (k+n) \text{ cd}^{\pi} \rightarrow i \rangle$  **and**  $\text{csi}: \langle \text{cs}^{\pi'} i' = \text{cs}^{\pi} i \rangle$  **and**  $\text{ik}: \langle i < k \rangle$  **using**  $\text{cdleswap}'$  **by**  
 $\text{force}$   
**have**  $\text{ij}: \langle i \leq j \rangle$  **using**  $\text{jge } \text{cdi } \text{ik}$  **by**  $\text{auto}$   
**show**  $\langle i' \leq j' \rangle$  **using**  $\text{cs-order-le}[OF \text{ assms}(1,2) \ \text{csi}[\text{symmetric}] \ \text{csj} - \text{ij}] \ \text{cd-not-ret}[OF \ \text{cdi}]$  **by**  $\text{simp}$   
**qed**  
**have**  $\langle \text{cs}^{\pi} (k + n) = \text{cs}^{\pi} j @ \text{cs}^{\pi} \ll k \ n \rangle$  **using**  $\text{cs-split-shift-cd}[OF \ \text{cdj } \text{jk} - \text{jge}] \ n0$  **by**  $\text{auto}$   
**moreover**  
**have**  $\langle \text{cs}^{\pi'} (k' + n') = \text{cs}^{\pi'} j' @ \text{cs}^{\pi'} \ll k' \ n' \rangle$  **using**  $\text{cs-split-shift-cd}[OF \ \text{cdj}' \ \text{jk}' - \text{jge}'] \ n0'$  **by**  $\text{auto}$   
**ultimately**  
**have**  $\langle \text{cs}^{\pi} (k+n) = \text{cs}^{\pi'} (k'+n') \rangle$  **using**  $\text{csj assms}(4)$  **by**  $\text{auto}$   
**thus**  $\langle \text{False} \rangle$  **using**  $\text{ne}$  **by**  $\text{simp}$   
**qed**

**lemma**  $\text{cs-eq-is-eq-shifted}$ : **assumes**  $\langle \text{is-path } \pi \rangle$  **and**  $\langle \text{is-path } \pi' \rangle$  **and**  $\langle \text{cs}^{\pi} k = \text{cs}^{\pi'} k' \rangle$  **and**  $\langle \text{cs}^{\pi} (k+n) = \text{cs}^{\pi'} (k'+n') \rangle$  **shows**  $\langle \text{cs}^{\pi} \ll k \ n = \text{cs}^{\pi'} \ll k' \ n' \rangle$   
**proof** ( $\text{rule } \text{ccontr}$ )  
**assume**  $\text{ne}: \langle \text{cs}^{\pi} \ll k \ n \neq \text{cs}^{\pi'} \ll k' \ n' \rangle$   
**have**  $\text{nretkn}: \langle \pi (k+n) \neq \text{return} \rangle$  **proof**  
**assume**  $1: \langle \pi (k+n) = \text{return} \rangle$   
**hence**  $2: \langle \pi' (k'+n') = \text{return} \rangle$  **using**  $\text{assms}(4) \ \text{last-cs}$  **by**  $\text{metis}$   
**hence**  $\langle (\pi \ll k) \ n = \text{return} \rangle \langle (\pi' \ll k') \ n' = \text{return} \rangle$  **using**  $1$  **by**  $\text{auto}$   
**hence**  $\langle \text{cs}^{\pi} \ll k \ n = \text{cs}^{\pi'} \ll k' \ n' \rangle$  **using**  $\text{cs-return}$  **by**  $\text{metis}$   
**thus**  $\langle \text{False} \rangle$  **using**  $\text{ne}$  **by**  $\text{simp}$   
**qed**  
**hence**  $\text{nretk}: \langle \pi k \neq \text{return} \rangle$  **using**  $\text{term-path-stable}[OF \ \text{assms}(1), \ \text{of } \langle k \rangle \langle k+n \rangle]$  **by**  $\text{auto}$   
**have**  $\text{nretkn}': \langle \pi' (k'+n') \neq \text{return} \rangle$  **proof**  
**assume**  $1: \langle \pi' (k'+n') = \text{return} \rangle$   
**hence**  $2: \langle \pi (k+n) = \text{return} \rangle$  **using**  $\text{assms}(4) \ \text{last-cs}$  **by**  $\text{metis}$   
**hence**  $\langle (\pi \ll k) \ n = \text{return} \rangle \langle (\pi' \ll k') \ n' = \text{return} \rangle$  **using**  $1$  **by**  $\text{auto}$   
**hence**  $\langle \text{cs}^{\pi} \ll k \ n = \text{cs}^{\pi'} \ll k' \ n' \rangle$  **using**  $\text{cs-return}$  **by**  $\text{metis}$   
**thus**  $\langle \text{False} \rangle$  **using**  $\text{ne}$  **by**  $\text{simp}$

qed

hence  $nretk'$ :  $\langle \pi' k' \neq return \rangle$  **using**  $term\text{-}path\text{-}stable[OF\ assms(2),\ of\ \langle k' \rangle\ \langle k' + n' \rangle]$  **by**  $auto$   
have  $n0$ :  $\langle n > 0 \rangle$  **proof** (rule  $ccontr$ )

assume \*:  $\langle \neg 0 < n \rangle$

hence  $\langle cs^{\pi'} k' = cs^{\pi'} (k' + n') \rangle$  **using**  $assms(3,4)$  **by**  $auto$

hence  $n0$ :  $\langle n = 0 \rangle\ \langle n' = 0 \rangle$  **using**  $cs\text{-}inj[OF\ assms(2)\ nretkn',\ of\ \langle k' \rangle]$  \* **by**  $auto$

have  $\langle cs^{\pi} \ll k\ n = cs^{\pi'} \ll k'\ n' \rangle$  **unfolding**  $n0\ cs\text{-}0$  **by** ( $auto$ ,  $metis\ last\text{-}cs\ assms(3)$ )

thus  $\langle False \rangle$  **using**  $ne$  **by**  $simp$

qed

have  $n0'$ :  $\langle n' > 0 \rangle$  **proof** (rule  $ccontr$ )

assume \*:  $\langle \neg 0 < n' \rangle$

hence  $\langle cs^{\pi} k = cs^{\pi} (k + n) \rangle$  **using**  $assms(3,4)$  **by**  $auto$

hence  $n0$ :  $\langle n = 0 \rangle\ \langle n' = 0 \rangle$  **using**  $cs\text{-}inj[OF\ assms(1)\ nretkn,\ of\ \langle k \rangle]$  \* **by**  $auto$

have  $\langle cs^{\pi} \ll k\ n = cs^{\pi'} \ll k'\ n' \rangle$  **unfolding**  $n0\ cs\text{-}0$  **by** ( $auto$ ,  $metis\ last\text{-}cs\ assms(3)$ )

thus  $\langle False \rangle$  **using**  $ne$  **by**  $simp$

qed

have  $cdle$ :  $\langle \exists j. (k+n)\ cd^{\pi} \rightarrow j \wedge j < k \rangle$  (is  $\langle \exists j. ?P\ j \rangle$ ) **proof** (rule  $ccontr$ )

assume  $\langle \neg (\exists j. (k+n)\ cd^{\pi} \rightarrow j \wedge j < k) \rangle$

hence  $allge$ :  $\langle \forall j. (k+n)\ cd^{\pi} \rightarrow j \longrightarrow k \leq j \rangle$  **by**  $auto$

have  $allge'$ :  $\langle \forall j'. (k'+n')\ cd^{\pi'} \rightarrow j' \longrightarrow k' \leq j' \rangle$  **proof** (rule, rule)

fix  $j'$

assume \*:  $\langle k' + n'\ cd^{\pi'} \rightarrow j' \rangle$

obtain  $j$  where  $cdj$ :  $\langle k+n\ cd^{\pi} \rightarrow j \rangle$  and  $csj$ :  $\langle cs^{\pi} j = cs^{\pi'} j' \rangle$  **using**  $cs\text{-}path\text{-}swap\text{-}cd[OF\ assms(2,1)\ assms(4)[symmetric]\ *]$  **by**  $metis$

hence  $kj$ :  $\langle k \leq j \rangle$  **using**  $allge$  **by**  $auto$

thus  $kj'$ :  $\langle k' \leq j' \rangle$  **using**  $cs\text{-}order\text{-}le[OF\ assms(1,2,3)\ csj\ nretk]$  **by**  $simp$

qed

have  $\langle cs^{\pi} (k + n) = cs^{\pi} \ll k\ n \rangle$  **using**  $cs\text{-}split\text{-}shift\text{-}nocd[OF\ assms(1)\ -\ allge]$   $n0$  **by**  $auto$

moreover

have  $\langle cs^{\pi'} (k' + n') = cs^{\pi'} \ll k'\ n' \rangle$  **using**  $cs\text{-}split\text{-}shift\text{-}nocd[OF\ assms(2)\ -\ allge']$   $n0'$  **by**  $auto$

ultimately

show  $\langle False \rangle$  **using**  $ne\ assms(4)$  **by**  $auto$

qed

define  $j$  where  $\langle j ==\ GREATEST\ j. (k+n)\ cd^{\pi} \rightarrow j \wedge j < k \rangle$

have  $cdj$ :  $\langle (k+n)\ cd^{\pi} \rightarrow j \rangle$  and  $jk$ :  $\langle j < k \rangle$  and  $jge$ :  $\langle \forall j' < k. (k+n)\ cd^{\pi} \rightarrow j' \longrightarrow j' \leq j \rangle$  **proof** -

have  $bound$ :  $\langle \forall y. ?P\ y \longrightarrow y \leq k \rangle$  **by**  $auto$

show  $\langle (k+n)\ cd^{\pi} \rightarrow j \rangle$  **using**  $GreatestI\text{-}nat[of\ \langle ?P \rangle]$   $bound\ j\text{-}def\ cdle$  **by**  $blast$

show  $\langle j < k \rangle$  **using**  $GreatestI\text{-}nat[of\ \langle ?P \rangle]$   $bound\ j\text{-}def\ cdle$  **by**  $blast$

show  $\langle \forall j' < k. (k+n)\ cd^{\pi} \rightarrow j' \longrightarrow j' \leq j \rangle$  **using**  $Greatest\text{-}le\text{-}nat[of\ \langle ?P \rangle]$   $bound\ j\text{-}def$  **by**  $blast$

qed

obtain  $j'$  where  $cdj'$ :  $\langle (k'+n')\ cd^{\pi'} \rightarrow j' \rangle$  and  $csj$ :  $\langle cs^{\pi} j = cs^{\pi'} j' \rangle$  **using**  $cs\text{-}path\text{-}swap\text{-}cd\ assms\ cdj$  **by**  $blast$

have  $jge'$ :  $\langle \forall i' < k'. (k'+n')\ cd^{\pi'} \rightarrow i' \longrightarrow i' \leq j' \rangle$  **proof**(rule,rule,rule)

fix  $i'$

assume  $ik'$ :  $\langle i' < k' \rangle$  and  $cdi'$ :  $\langle k' + n'\ cd^{\pi'} \rightarrow i' \rangle$

then obtain  $i$  where  $cdi$ :  $\langle (k+n)\ cd^{\pi} \rightarrow i \rangle$  and  $csi$ :  $\langle cs^{\pi'} i' = cs^{\pi} i \rangle$  **using**  $cs\text{-}path\text{-}swap\text{-}cd[OF\ assms(2,1)\ assms(4)[symmetric]]$  **by**  $blast$

have  $nreti'$ :  $\langle \pi' i' \neq return \rangle$  **by** ( $metis\ cd\text{-}not\text{-}ret\ cdi'$ )

have  $ik$ :  $\langle i < k \rangle$  **using**  $cs\text{-}order[OF\ assms(2,1)\ csi\ -\ nreti'\ ik']$   $assms(3)$  **by**  $auto$

have  $ij$ :  $\langle i \leq j \rangle$  **using**  $jge\ cdi\ ik$  **by**  $auto$

show  $\langle i' \leq j' \rangle$  **using**  $cs\text{-}order\text{-}le[OF\ assms(1,2)\ csi[symmetric]\ csj\ -\ ij]$   $cd\text{-}not\text{-}ret[OF\ cdi]$  **by**  $simp$

qed

have  $jk'$ :  $\langle j' < k' \rangle$  **using**  $cs\text{-}order[OF\ assms(1,2)\ csj\ assms(3)\ cd\text{-}not\text{-}ret[OF\ cdj]\ jk]$  .

have  $\langle cs^{\pi} (k + n) = cs^{\pi} j @\ cs^{\pi} \ll k\ n \rangle$  **using**  $cs\text{-}split\text{-}shift\text{-}cd[OF\ cdj\ jk\ -\ jge]$   $n0$  **by**  $auto$

moreover

**have**  $\langle cs^{\pi'}(k' + n') = cs^{\pi'} j' @ cs^{\pi'} \ll k' n' \rangle$  **using** *cs-split-shift-cd*[*OF cdj' jk' - jge'*] *n0'* **by auto**  
**ultimately**  
**have**  $\langle cs^{\pi} \ll k n = cs^{\pi'} \ll k' n' \rangle$  **using** *csj assms(4)* **by auto**  
**thus**  $\langle False \rangle$  **using** *ne* **by simp**  
**qed**

**lemma** *converged-cd-diverge-cs*: **assumes**  $\langle is\text{-path } \pi \rangle$  **and**  $\langle is\text{-path } \pi' \rangle$  **and**  $\langle cs^{\pi} j = cs^{\pi'} j' \rangle$  **and**  $\langle j < l \rangle$  **and**  
 $\langle \neg (\exists l'. cs^{\pi} l = cs^{\pi'} l') \rangle$  **and**  $\langle l < m \rangle$  **and**  $\langle cs^{\pi} m = cs^{\pi'} m' \rangle$

**obtains**  $k k'$  **where**  $\langle j \leq k \rangle$   $\langle cs^{\pi} k = cs^{\pi'} k' \rangle$  **and**  $\langle l \text{ cd}^{\pi} \rightarrow k \rangle$  **and**  $\langle \pi (Suc k) \neq \pi' (Suc k') \rangle$

**proof** –

**have**  $\langle is\text{-path } (\pi \ll j) \rangle$   $\langle is\text{-path } (\pi' \ll j') \rangle$  **using** *assms(1,2) path-path-shift* **by auto**

**moreover**

**have**  $\langle (\pi \ll j) 0 = (\pi' \ll j') 0 \rangle$  **using** *assms(3) last-cs* **by** (*metis path-shift-def add.right-neutral*)

**moreover**

**have**  $\langle \neg (\exists l'. cs^{\pi} \ll j (l - j) = cs^{\pi'} \ll j' l') \rangle$  **proof**

**assume**  $\langle \exists l'. cs^{\pi} \ll j (l - j) = cs^{\pi'} \ll j' l' \rangle$

**then obtain**  $l'$  **where** *csl*:  $\langle cs^{\pi} \ll j (l - j) = cs^{\pi'} \ll j' l' \rangle$  **by blast**

**have**  $\langle cs^{\pi} l = cs^{\pi'} (j' + l') \rangle$  **using** *shifted-cs-eq-is-eq*[*OF assms(1,2,3) csl*] *assms(4)* **by auto**

**thus**  $\langle False \rangle$  **using** *assms(5)* **by blast**

**qed**

**moreover**

**have**  $\langle l - j < m - j \rangle$  **using** *assms* **by auto**

**moreover**

**have**  $\langle \pi j \neq \text{return} \rangle$  **using** *cs-return assms(1-5) term-path-stable* **by** (*metis nat-less-le*)

**hence**  $\langle j' < m' \rangle$  **using** *cs-order*[*OF assms(1,2,3,7)*] *assms* **by auto**

**hence**  $\langle cs^{\pi} \ll j (m - j) = cs^{\pi'} \ll j' (m' - j') \rangle$  **using** *cs-eq-is-eq-shifted*[*OF assms(1,2,3), of*  $\langle m - j \rangle$   $\langle m' - j' \rangle$ ] *assms(4,6,7)*

**by auto**

**ultimately**

**obtain**  $k k'$  **where** *csk*:  $\langle cs^{\pi} \ll j k = cs^{\pi'} \ll j' k' \rangle$  **and** *lck*:  $\langle l - j \text{ cd}^{\pi} \ll j \rightarrow k \rangle$  **and** *suc*:  $\langle (\pi \ll j) (Suc k) \neq (\pi' \ll j') (Suc k') \rangle$  **using** *converged-cd-diverge* **by blast**

**have**  $\langle cs^{\pi} (j + k) = cs^{\pi'} (j' + k') \rangle$  **using** *shifted-cs-eq-is-eq*[*OF assms(1-3) csk*] .

**moreover**

**have**  $\langle l \text{ cd}^{\pi} \rightarrow j + k \rangle$  **using** *lck assms(1,2,4)* **by** (*metis add.commute add-diff-cancel-right' cd-path-shift le-add1*)

**moreover**

**have**  $\langle \pi (Suc (j + k)) \neq \pi' (Suc (j' + k')) \rangle$  **using** *suc* **by auto**

**moreover**

**have**  $\langle j \leq j + k \rangle$  **by auto**

**ultimately**

**show**  $\langle thesis \rangle$  **using** *that*[*of*  $\langle j + k \rangle$   $\langle j' + k' \rangle$ ] **by auto**

**qed**

**lemma** *cs-ipd-conv*: **assumes** *csk*:  $\langle cs^{\pi} k = cs^{\pi'} k' \rangle$  **and** *ipd*:  $\langle \pi l = ipd (\pi k) \rangle$   $\langle \pi' l' = ipd (\pi' k') \rangle$

**and** *nipd*:  $\langle \forall n \in \{k..<l\}. \pi n \neq ipd (\pi k) \rangle$   $\langle \forall n' \in \{k'..<l'\}. \pi' n' \neq ipd (\pi' k') \rangle$  **and** *kl*:  $\langle k < l \rangle$   $\langle k' < l' \rangle$

**shows**  $\langle cs^{\pi} l = cs^{\pi'} l' \rangle$  **using** *cs-ipd*[*OF ipd(1) nipd(1) kl(1)*] *cs-ipd*[*OF ipd(2) nipd(2) kl(2)*] *csk ipd* **by** (*metis (no-types) last-cs*)

**lemma** *cp-eq-cs*: **assumes**  $\langle ((\sigma, k), (\sigma', k')) \in cp \rangle$  **shows**  $\langle cs^{\text{path } \sigma} k = cs^{\text{path } \sigma'} k' \rangle$

**using** *assms*

**apply** (*induction rule: cp.induct*)

**apply** *blast+*



apply simp  
done

**lemma** *cd-cs-swap*: **assumes**  $\langle l \text{ cd}^\pi \rightarrow k \rangle \langle \text{cs}^\pi l = \text{cs}^{\pi'} l' \rangle \langle \text{cs}^\pi k = \text{cs}^{\pi'} k' \rangle$  **shows**  $\langle l' \text{ cd}^{\pi'} \rightarrow k' \rangle$  **proof** –  
**have**  $\langle \exists i. l \text{ icd}^\pi \rightarrow i \rangle$  **using** *assms(1) excd-impl-exicd* **by** *blast*  
**hence**  $\langle \text{cs}^\pi l \neq [\pi l] \rangle$  **by** *auto*  
**hence**  $\langle \text{cs}^{\pi'} l' \neq [\pi' l'] \rangle$  **using** *assms last-cs* **by** *metis*  
**hence**  $\langle \exists i'. l' \text{ icd}^{\pi'} \rightarrow i' \rangle$  **by** (*metis cs-cases*)  
**hence** *path'*:  $\langle \text{is-path } \pi' \rangle$  **unfolding** *is-icdi-def is-cdi-def* **by** *auto*  
**from** *cd-in-cs[OF assms(1)]*  
**obtain** *ys* **where** *csl*:  $\langle \text{cs}^\pi l = \text{cs}^\pi k @ \text{ys} @ [\pi l] \rangle$  **by** *blast*  
**obtain** *xs* **where** *csk*:  $\langle \text{cs}^\pi k = \text{xs} @ [\pi k] \rangle$  **by** (*metis append-butlast-last-id cs-not-nil last-cs*)  
**have**  $\langle \pi l = \pi' l' \rangle$  **using** *assms last-cs* **by** *metis*  
**have** *csl'*:  $\langle \text{cs}^{\pi'} l' = \text{xs} @ [\pi k] @ \text{ys} @ [\pi' l'] \rangle$  **by** (*metis*  $\pi l$  *append-eq-appendI assms(2) csk csl*)  
**from** *cs-split[of  $\langle \pi' \rangle \langle l' \rangle \langle \text{xs} \rangle \langle \pi k \rangle \langle \text{ys} \rangle$ ]*  
**obtain** *m* **where** *csm*:  $\langle \text{cs}^{\pi'} m = \text{xs} @ [\pi k] \rangle$  **and** *lcm*:  $\langle l' \text{ cd}^{\pi'} \rightarrow m \rangle$  **using** *csl'* **by** *metis*  
**have** *csm'*:  $\langle \text{cs}^{\pi'} m = \text{cs}^{\pi'} k' \rangle$  **by** (*metis assms(3) csk csm*)  
**have**  $\langle \pi' m \neq \text{return} \rangle$  **using** *lcm* **unfolding** *is-cdi-def* **using** *term-path-stable* **by** (*metis nat-less-le*)  
**hence**  $\langle m = k' \rangle$  **using** *cs-inj path' csm'* **by** *auto*  
**thus**  $\langle ?thesis \rangle$  **using** *lcm* **by** *auto*  
**qed**

## 2.6 Facts about Observations

**lemma** *kth-obs-not-none*: **assumes**  $\langle \text{is-kth-obs } (\text{path } \sigma) k i \rangle$  **obtains** *a* **where**  $\langle \text{obs } \sigma i = \text{Some } a \rangle$  **using** *assms* **unfolding** *is-kth-obs-def obsp-def* **by** *auto*

**lemma** *kth-obs-unique*:  $\langle \text{is-kth-obs } \pi k i \implies \text{is-kth-obs } \pi k j \implies i = j \rangle$  **proof** (*induction*  $\langle i \rangle \langle j \rangle$  *rule*: *nat-sym-cases*)

**case** *sym* **thus**  $\langle ?case \rangle$  **by** *simp*  
**next**  
**case** *eq* **thus**  $\langle ?case \rangle$  **by** *simp*  
**next**  
**case** (*less i j*)  
**have**  $\langle \text{obs-ids } \pi \cap \{..<i\} \subseteq \text{obs-ids } \pi \cap \{..<j\} \rangle$  **using** *less(1)* **by** *auto*  
**moreover**  
**have**  $\langle i \in \text{obs-ids } \pi \cap \{..<j\} \rangle$  **using** *less* **unfolding** *is-kth-obs-def obs-ids-def* **by** *auto*  
**moreover**  
**have**  $\langle i \notin \text{obs-ids } \pi \cap \{..<i\} \rangle$  **by** *auto*  
**moreover**  
**have**  $\langle \text{card } (\text{obs-ids } \pi \cap \{..<i\}) = \text{card } (\text{obs-ids } \pi \cap \{..<j\}) \rangle$  **using** *less.prem*s **unfolding** *is-kth-obs-def* **by** *auto*  
**moreover**  
**have**  $\langle \text{finite } (\text{obs-ids } \pi \cap \{..<i\}) \rangle \langle \text{finite } (\text{obs-ids } \pi \cap \{..<j\}) \rangle$  **by** *auto*  
**ultimately**  
**have**  $\langle \text{False} \rangle$  **by** (*metis card-subset-eq*)  
**thus**  $\langle ?case \rangle$  ..  
**qed**

**lemma** *obs-none-no-kth-obs*: **assumes**  $\langle \text{obs } \sigma k = \text{None} \rangle$  **shows**  $\langle \neg (\exists i. \text{is-kth-obs } (\text{path } \sigma) k i) \rangle$   
**apply** *rule*  
**using** *assms*  
**unfolding** *obs-def obsp-def*  
**apply** (*auto split: option.split-asm*)  
**by** (*metis assms kth-obs-not-none kth-obs-unique obs-def option.distinct(2) the-equality*)

**lemma** *obs-some-kth-obs* : **assumes**  $\langle \text{obs } \sigma \ k \neq \text{None} \rangle$  **obtains**  $i$  **where**  $\langle \text{is-kth-obs } (\text{path } \sigma) \ k \ i \rangle$  **by** (*metis obs-def assms*)

**lemma** *not-none-is-obs*: **assumes**  $\langle \text{att}(\pi \ i) \neq \text{None} \rangle$  **shows**  $\langle \text{is-kth-obs } \pi \ (\text{card } (\text{obs-ids } \pi \cap \{..<i\})) \ i \rangle$  **unfolding** *is-kth-obs-def* **using** *assms* **by** *auto*

**lemma** *in-obs-ids-is-kth-obs*: **assumes**  $\langle i \in \text{obs-ids } \pi \rangle$  **obtains**  $k$  **where**  $\langle \text{is-kth-obs } \pi \ k \ i \rangle$  **proof**  
**have**  $\langle \text{att } (\pi \ i) \neq \text{None} \rangle$  **using** *assms obs-ids-def* **by** *auto*  
**thus**  $\langle \text{is-kth-obs } \pi \ (\text{card } (\text{obs-ids } \pi \cap \{..<i\})) \ i \rangle$  **using** *not-none-is-obs* **by** *auto*  
**qed**

**lemma** *kth-obs-stable*: **assumes**  $\langle \text{is-kth-obs } \pi \ l \ j \rangle \langle k < l \rangle$  **shows**  $\langle \exists \ i. \text{is-kth-obs } \pi \ k \ i \rangle$  **using** *assms* **proof**  
(*induction*  $\langle l \rangle$  *arbitrary*:  $\langle j \rangle$  *rule*: *less-induct*)

**case** (*less*  $l \ j$ )  
**have** *cardl*:  $\langle \text{card } (\text{obs-ids } \pi \cap \{..<j\}) = l \rangle$  **using** *less is-kth-obs-def* **by** *auto*  
**then obtain**  $i$  **where** *ex*:  $\langle i \in \text{obs-ids } \pi \cap \{..<j\} \rangle$  (*is*  $\langle ?P \ i \rangle$ ) **using** *less(3)* **by** (*metis card.empty empty-iff less-irrefl subsetI subset-antisym zero-diff zero-less-diff*)  
**have** *bound*:  $\langle \forall \ i. i \in \text{obs-ids } \pi \cap \{..<j\} \longrightarrow i \leq j \rangle$  **by** *auto*  
**let**  $\langle ?i \rangle = \langle \text{GREATEST } i. i \in \text{obs-ids } \pi \cap \{..<j\} \rangle$   
**have**  $*$ :  $\langle ?i < j \rangle \langle ?i \in \text{obs-ids } \pi \rangle$  **using** *GreatestI-nat*[*of*  $\langle ?P \rangle \langle i \rangle \langle j \rangle$ ] *ex bound* **by** *auto*  
**have**  $**$ :  $\langle \forall \ i. i \in \text{obs-ids } \pi \wedge i < j \longrightarrow i \leq ?i \rangle$  **using** *Greatest-le-nat*[*of*  $\langle ?P \rangle - \langle j \rangle$ ] *ex bound* **by** *auto*  
**have**  $\langle (\text{obs-ids } \pi \cap \{..<?i\}) \cup \{?i\} = \text{obs-ids } \pi \cap \{..<j\} \rangle$  **apply rule** **apply** *auto* **using**  $*$ [*simplified*] **apply** *simp+* **using**  $**$ [*simplified*] **by** *auto*

**moreover**  
**have**  $\langle ?i \notin (\text{obs-ids } \pi \cap \{..<?i\}) \rangle$  **by** *auto*  
**ultimately**  
**have**  $\langle \text{Suc } (\text{card } (\text{obs-ids } \pi \cap \{..<?i\})) = l \rangle$  **using** *cardl* **by** (*metis Un-empty-right Un-insert-right card-insert-disjoint finite-Int finite-lessThan*)

**hence**  $\langle \text{card } (\text{obs-ids } \pi \cap \{..<?i\}) = l - 1 \rangle$  **by** *auto*  
**hence** *iko*:  $\langle \text{is-kth-obs } \pi \ (l - 1) \ ?i \rangle$  **using**  $*$ (2) **unfolding** *is-kth-obs-def obs-ids-def* **by** *auto*  
**have** *ll*:  $\langle l - 1 < l \rangle$  **by** (*metis One-nat-def diff-Suc-less less.premis(2) not-gr0 not-less0*)  
**note** *IV=less(1)*[*OF ll iko*]  
**show**  $\langle ?thesis \rangle$  **proof cases**  
**assume**  $\langle k < l - 1 \rangle$  **thus**  $\langle ?thesis \rangle$  **using** *IV* **by** *simp*  
**next**  
**assume**  $\langle \neg k < l - 1 \rangle$   
**hence**  $\langle k = l - 1 \rangle$  **using** *less* **by** *auto*  
**thus**  $\langle ?thesis \rangle$  **using** *iko* **by** *blast*

**qed**  
**qed**

**lemma** *kth-obs-mono*: **assumes**  $\langle \text{is-kth-obs } \pi \ k \ i \rangle \langle \text{is-kth-obs } \pi \ l \ j \rangle \langle k < l \rangle$  **shows**  $\langle i < j \rangle$  **proof** (*rule ccontr*)

**assume**  $\langle \neg i < j \rangle$   
**hence**  $\langle \{..<j\} \subseteq \{..<i\} \rangle$  **by** *auto*  
**hence**  $\langle \text{obs-ids } \pi \cap \{..<j\} \subseteq \text{obs-ids } \pi \cap \{..<i\} \rangle$  **by** *auto*  
**moreover**  
**have**  $\langle \text{finite } (\text{obs-ids } \pi \cap \{..<i\}) \rangle$  **by** *auto*  
**ultimately**  
**have**  $\langle \text{card } (\text{obs-ids } \pi \cap \{..<j\}) \leq \text{card } (\text{obs-ids } \pi \cap \{..<i\}) \rangle$  **by** (*metis card-mono*)  
**thus**  $\langle \text{False} \rangle$  **using** *assms* **unfolding** *is-kth-obs-def* **by** *auto*

**qed**

**lemma** *kth-obs-le-iff*: **assumes**  $\langle \text{is-kth-obs } \pi \ k \ i \rangle \langle \text{is-kth-obs } \pi \ l \ j \rangle$  **shows**  $\langle k < l \iff i < j \rangle$  **by** (*metis assms kth-obs-unique kth-obs-mono not-less-iff-gr-or-eq*)

**lemma** *ret-obs-all-obs*: **assumes** *path*:  $\langle is\text{-path } \pi \rangle$  **and** *iki*:  $\langle is\text{-kth-obs } \pi k i \rangle$  **and** *ret*:  $\langle \pi i = return \rangle$  **and** *kl*:  $\langle k < l \rangle$  **obtains** *j* **where**  $\langle is\text{-kth-obs } \pi l j \rangle$

**proof**—

**show**  $\langle thesis \rangle$

**using** *kl iki ret* **proof** (*induction*  $\langle l - k \rangle$  *arbitrary*:  $\langle k \rangle \langle i \rangle$  *rule*: *less-induct*)

**case** (*less k i*)

**note**  $kl = \langle k < l \rangle$

**note**  $iki = \langle is\text{-kth-obs } \pi k i \rangle$

**note**  $ret = \langle \pi i = return \rangle$

**have** *card*:  $\langle card (obs\text{-ids } \pi \cap \{..<i\}) = k \rangle$  **and** *att-ret*:  $\langle att\ return \neq None \rangle$  **using** *iki ret* **unfolding** *is-kth-obs-def* **by** *auto*

**have** *rets*:  $\langle \pi (Suc\ i) = return \rangle$  **using** *path ret term-path-stable* **by** *auto*

**hence** *attsuc*:  $\langle att (\pi (Suc\ i)) \neq None \rangle$  **using** *att-ret* **by** *auto*

**hence**  $*$ :  $\langle i \in obs\text{-ids } \pi \rangle$  **using** *att-ret ret* **unfolding** *obs-ids-def* **by** *auto*

**have**  $\langle \{..< Suc\ i\} = insert\ i\ \{..<i\} \rangle$  **by** *auto*

**hence** *a*:  $\langle obs\text{-ids } \pi \cap \{..< Suc\ i\} = insert\ i\ (obs\text{-ids } \pi \cap \{..<i\}) \rangle$  **using**  $*$  **by** *auto*

**have** *b*:  $\langle i \notin obs\text{-ids } \pi \cap \{..<i\} \rangle$  **by** *auto*

**have**  $\langle finite (obs\text{-ids } \pi \cap \{..<i\}) \rangle$  **by** *auto*

**hence**  $\langle card (obs\text{-ids } \pi \cap \{..< Suc\ i\}) = Suc\ k \rangle$  **by** (*metis card card-insert-disjoint a b*)

**hence** *iksuc*:  $\langle is\text{-kth-obs } \pi (Suc\ k) (Suc\ i) \rangle$  **using** *attsuc* **unfolding** *is-kth-obs-def* **by** *auto*

**have** *suckl*:  $\langle Suc\ k \leq l \rangle$  **using** *kl* **by** *auto*

**note** *less*

**thus**  $\langle thesis \rangle$  **proof** (*cases*  $\langle Suc\ k < l \rangle$ )

**assume** *skl*:  $\langle Suc\ k < l \rangle$

**from** *less(1)[OF - skl iksuc rets]* *skl*

**show**  $\langle thesis \rangle$  **by** *auto*

**next**

**assume**  $\langle \neg Suc\ k < l \rangle$

**hence**  $\langle Suc\ k = l \rangle$  **using** *suckl* **by** *auto*

**thus**  $\langle thesis \rangle$  **using** *iksuc that* **by** *auto*

**qed**

**qed**

**qed**

**lemma** *no-kth-obs-missing-cs*: **assumes** *path*:  $\langle is\text{-path } \pi \rangle \langle is\text{-path } \pi' \rangle$  **and** *iki*:  $\langle is\text{-kth-obs } \pi k i \rangle$  **and** *not-in- $\pi'$* :  $\langle \neg (\exists i'. is\text{-kth-obs } \pi' k i') \rangle$  **obtains** *l j* **where**  $\langle is\text{-kth-obs } \pi l j \rangle \langle \neg (\exists j'. cs^\pi j = cs^{\pi'} j') \rangle$

**proof** (*rule ccontr*)

**assume**  $\langle \neg thesis \rangle$

**hence** *all-in- $\pi'$* :  $\langle \forall l j. is\text{-kth-obs } \pi l j \longrightarrow (\exists j'. cs^\pi j = cs^{\pi'} j') \rangle$  **using** *that* **by** *blast*

**then obtain** *i'* **where** *csi*:  $\langle cs^\pi i = cs^{\pi'} i' \rangle$  **using** *assms* **by** *blast*

**hence**  $\langle att(\pi' i') \neq None \rangle$  **using** *iki* **by** (*metis is-kth-obs-def last-cs*)

**then obtain** *k'* **where** *ik'*:  $\langle is\text{-kth-obs } \pi' k' i' \rangle$  **by** (*metis not-none-is-obs*)

**hence** *kk'*:  $\langle k' < k \rangle$  **using** *not-in- $\pi'$  kth-obs-stable* **by** (*auto, metis not-less-iff-gr-or-eq*)

**show**  $\langle False \rangle$  **proof** (*cases*  $\langle \pi i = return \rangle$ )

**assume**  $\langle \pi i \neq return \rangle$

**thus**  $\langle False \rangle$  **using** *kk' ik' csi iki* **proof** (*induction*  $\langle k \rangle$  *arbitrary*:  $\langle i \rangle \langle i' \rangle \langle k' \rangle$ )

**case** 0 **thus**  $\langle ?case \rangle$  **by** *simp*

**next**

**case** (*Suc k i i' k'*)

**then obtain** *j* **where** *ikj*:  $\langle is\text{-kth-obs } \pi k j \rangle$  **by** (*metis kth-obs-stable lessI*)

**then obtain** *j'* **where** *csj*:  $\langle cs^\pi j = cs^{\pi'} j' \rangle$  **using** *all-in- $\pi'$*  **by** *blast*

**hence**  $\langle att(\pi' j') \neq None \rangle$  **using** *ikj* **by** (*metis is-kth-obs-def last-cs*)

**then obtain** *k2* **where** *ik2*:  $\langle is\text{-kth-obs } \pi' k2 j' \rangle$  **by** (*metis not-none-is-obs*)

**have** *ji*:  $\langle j < i \rangle$  **using** *kth-obs-mono [OF ikj is-kth-obs  $\pi (Suc\ k) i$ ]* **by** *auto*

**hence** *nretj*:  $\langle \pi j \neq return \rangle$  **using** *Suc(2) term-path-stable less-imp-le path(1)* **by** *metis*

**have**  $ji'$ :  $\langle j' < i' \rangle$  **using**  $cs\text{-order}[OF\ path\ \text{-}\ nretj, of\ \langle j' \rangle\ \langle i' \rangle\ \langle i' \rangle]$   $csj\ \langle cs^\pi\ i = cs^{\pi'}\ i' \rangle$   $ji$  **by**  $auto$   
**have**  $\langle k2 \neq k' \rangle$  **using**  $ik2\ Suc(4)$   $ji'$   $kth\text{-obs}\text{-unique}[of\ \langle \pi' \rangle\ \langle k' \rangle\ \langle i' \rangle\ \langle j' \rangle]$  **by**  $(metis\ less\ irrefl)$   
**hence**  $k2k'$ :  $\langle k2 < k' \rangle$  **using**  $kth\text{-obs}\text{-mono}[OF\ \langle is\text{-kth}\text{-obs}\ \pi'\ k'\ i' \rangle\ ik2]$   $ji'$  **by**  $(metis\ not\ less\ iff\ gr\ or\ eq)$   
**hence**  $k2k$ :  $\langle k2 < k \rangle$  **using**  $Suc$  **by**  $auto$   
**from**  $Suc.IH[OF\ nretj\ k2k\ ik2\ csj\ ikj]$  **show**  $\langle False \rangle$  .  
**qed**  
**next**  
**assume**  $\langle \pi\ i = return \rangle$   
**hence**  $reti'$ :  $\langle \pi'\ i' = return \rangle$  **by**  $(metis\ csi\ last\ cs)$   
**from**  $ret\text{-obs}\text{-all}\text{-obs}[OF\ path(2)\ ik'\ reti'\ kk', of\ \langle False \rangle]$   $not\text{-in}\text{-}\pi'$   
**show**  $\langle False \rangle$  **by**  $blast$   
**qed**  
**qed**

**lemma**  $kth\text{-obs}\text{-cs}\text{-missing}\text{-cs}$ : **assumes**  $path$ :  $\langle is\text{-path}\ \pi \rangle\ \langle is\text{-path}\ \pi' \rangle$  **and**  $iki$ :  $\langle is\text{-kth}\text{-obs}\ \pi\ k\ i \rangle$  **and**  $iki'$ :  $\langle is\text{-kth}\text{-obs}\ \pi'\ k\ i' \rangle$  **and**  $csi$ :  $\langle cs^\pi\ i \neq cs^{\pi'}\ i' \rangle$   
**obtains**  $l\ j$  **where**  $\langle j \leq i \rangle\ \langle is\text{-kth}\text{-obs}\ \pi\ l\ j \rangle\ \langle \neg (\exists j'. cs^\pi\ j = cs^{\pi'}\ j') \rangle$  |  $l'\ j'$  **where**  $\langle j' \leq i' \rangle\ \langle is\text{-kth}\text{-obs}\ \pi'\ l'\ j' \rangle\ \langle \neg (\exists j. cs^\pi\ j = cs^{\pi'}\ j') \rangle$   
**proof**  $(rule\ ccontr)$   
**assume**  $nt$ :  $\langle \neg\ thesis \rangle$   
**show**  $\langle False \rangle$  **using**  $iki\ iki'$   $csi$  **that** **proof**  $(induction\ \langle k \rangle\ arbitrary: \langle i \rangle\ \langle i' \rangle\ rule: less\ induct)$   
**case**  $(less\ k\ i\ i')$   
**hence**  $all\text{-in}\text{-}\pi'$ :  $\langle \forall l\ j. j \leq i \wedge is\text{-kth}\text{-obs}\ \pi\ l\ j \implies (\exists j'. cs^\pi\ j = cs^{\pi'}\ j') \rangle$   
**and**  $all\text{-in}\text{-}\pi$ :  $\langle \forall l'\ j'. j' \leq i' \wedge is\text{-kth}\text{-obs}\ \pi'\ l'\ j' \implies (\exists j. cs^\pi\ j = cs^{\pi'}\ j') \rangle$  **by**  $(metis\ nt)$   $(metis\ nt\ less(6))$   
**obtain**  $j\ j'$  **where**  $csji$ :  $\langle cs^\pi\ j = cs^{\pi'}\ i' \rangle$  **and**  $csij$ :  $\langle cs^{\pi'}\ i = cs^\pi\ j' \rangle$  **using**  $all\text{-in}\text{-}\pi\ all\text{-in}\text{-}\pi'$   $less$  **by**  $blast$   
**then** **obtain**  $l\ l'$  **where**  $ilj$ :  $\langle is\text{-kth}\text{-obs}\ \pi\ l\ j \rangle$  **and**  $ilj'$ :  $\langle is\text{-kth}\text{-obs}\ \pi'\ l'\ j' \rangle$  **by**  $(metis\ is\text{-kth}\text{-obs}\text{-def}\ last\ cs\ less.prem(1,2))$   
**have**  $lnk$ :  $\langle l \neq k \rangle$  **using**  $ilj\ csji\ less(2)\ less(4)\ kth\text{-obs}\text{-unique}$  **by**  $auto$   
**have**  $lnk'$ :  $\langle l' \neq k' \rangle$  **using**  $ilj'\ csij\ less(3)\ less(4)\ kth\text{-obs}\text{-unique}$  **by**  $auto$   
**have**  $cseq$ :  $\langle \forall l\ j\ j'. l < k \wedge is\text{-kth}\text{-obs}\ \pi\ l\ j \wedge is\text{-kth}\text{-obs}\ \pi'\ l'\ j' \implies cs^\pi\ j = cs^{\pi'}\ j' \rangle$  **proof** –  
**{** **fix**  $t\ p\ p'$  **assume**  $tk$ :  $\langle t < k \rangle$  **and**  $ikp$ :  $\langle is\text{-kth}\text{-obs}\ \pi\ t\ p \rangle$  **and**  $ikp'$ :  $\langle is\text{-kth}\text{-obs}\ \pi'\ t\ p' \rangle$   
**hence**  $pi$ :  $\langle p < i \rangle$  **and**  $pi'$ :  $\langle p' < i' \rangle$  **by**  $(metis\ kth\text{-obs}\text{-mono}\ less.prem(1))$   $(metis\ kth\text{-obs}\text{-mono}\ less.prem(2)\ tk\ ikp')$   
**have**  $*$ :  $\langle \bigwedge j\ l. j \leq p \implies is\text{-kth}\text{-obs}\ \pi\ l\ j \implies \exists j'. cs^\pi\ j = cs^{\pi'}\ j' \rangle$  **using**  $pi\ all\text{-in}\text{-}\pi'$  **by**  $auto$   
**have**  $**$ :  $\langle \bigwedge j'\ l'. j' \leq p' \implies is\text{-kth}\text{-obs}\ \pi'\ l'\ j' \implies \exists j. cs^\pi\ j = cs^{\pi'}\ j' \rangle$  **using**  $pi'\ all\text{-in}\text{-}\pi$  **by**  $auto$   
**have**  $\langle cs^\pi\ p = cs^{\pi'}\ p' \rangle$  **apply** $(rule\ ccontr)$  **using**  $less(1)[OF\ tk\ ikp\ ikp']\ **$  **by**  $blast$   
**}**  
**thus**  $\langle ?thesis \rangle$  **by**  $blast$   
**qed**  
**have**  $i'nret$ :  $\langle \pi\ i \neq return \vee \pi'\ i' \neq return \rangle$  **using**  $less\ cs\text{-return}$  **by**  $auto$   
**have**  $a$ :  $\langle k < l \vee k < l' \rangle$  **proof**  $(rule\ ccontr)$   
**assume**  $\langle \neg(k < l \vee k < l') \rangle$   
**hence**  $*$ :  $\langle l < k \rangle\ \langle l' < k' \rangle$  **using**  $lnk\ lnk'$  **by**  $auto$   
**hence**  $ji$ :  $\langle j < i \rangle$  **and**  $ji'$ :  $\langle j' < i' \rangle$  **using**  $ilj\ ilj'\ less(2,3)\ kth\text{-obs}\text{-mono}$  **by**  $auto$   
**show**  $\langle False \rangle$  **using**  $i'nret$  **proof**  
**assume**  $nreti$ :  $\langle \pi\ i \neq return \rangle$   
**hence**  $nretj'$ :  $\langle \pi'\ j' \neq return \rangle$  **using**  $last\ cs\ csij$  **by**  $metis$   
**show**  $\langle False \rangle$  **using**  $cs\text{-order}[OF\ path(2,1)\ csij[symmetric]\ csji[symmetric]\ nretj'\ ji']$   $ji$  **by**  $simp$   
**next**  
**assume**  $nreti'$ :  $\langle \pi'\ i' \neq return \rangle$   
**hence**  $nretj'$ :  $\langle \pi\ j \neq return \rangle$  **using**  $last\ cs\ csji$  **by**  $metis$   
**show**  $\langle False \rangle$  **using**  $cs\text{-order}[OF\ path\ csji\ csij\ nretj'\ ji']$   $ji'$  **by**  $simp$   
**qed**  
**qed**

**have**  $\langle l < k \vee l' < k \rangle$  **proof** (rule ccontr)  
**assume**  $\langle \neg (l < k \vee l' < k) \rangle$   
**hence**  $\langle k < l \rangle \langle k < l' \rangle$  **using** *lnk lnk'* **by** *auto*  
**hence**  $ji: \langle i < j \rangle$  **and**  $ji': \langle i' < j' \rangle$  **using** *ilj ilj' less(2,3) kth-obs-mono* **by** *auto*  
**show**  $\langle False \rangle$  **using** *ii'nret* **proof**  
**assume**  $nreti: \langle \pi i \neq return \rangle$   
**show**  $\langle False \rangle$  **using** *cs-order[OF path csij csji nreti ji] ji'* **by** *simp*  
**next**  
**assume**  $nreti': \langle \pi i' \neq return \rangle$   
**show**  $\langle False \rangle$  **using** *cs-order[OF path(2,1) csji[symmetric] csij[symmetric] nreti' ji'] ji* **by** *simp*  
**qed**  
**qed**  
**hence**  $\langle k < l \wedge l' < k \vee k < l' \wedge l < k \rangle$  **using** *a* **by** *auto*  
**thus**  $\langle False \rangle$  **proof**  
**assume**  $\langle k < l \wedge l' < k \rangle$   
**hence**  $kl: \langle k < l \rangle$  **and**  $lk': \langle l' < k \rangle$  **by** *auto*  
**hence**  $ij: \langle i < j \rangle$  **and**  $ji': \langle j' < i' \rangle$  **using** *less(2,3) ilj ilj' kth-obs-mono* **by** *auto*  
**have**  $nreti: \langle \pi i \neq return \rangle$  **by** (*metis csji ii'nret ij last-cs path(1) term-path-stable less-imp-le*)  
**obtain**  $h$  **where**  $ilh: \langle is-kth-obs \pi l' h \rangle$  **using** *ji' all-in- $\pi$  ilj' no-kth-obs-missing-cs path(1) path(2)* **by**  
(*metis kl lk' ilj kth-obs-stable*)  
**hence**  $\langle cs^\pi h = cs^\pi j' \rangle$  **using** *cseq lk' ilj'* **by** *blast*  
**hence**  $\langle cs^\pi i = cs^\pi h \rangle$  **using** *csij* **by** *auto*  
**hence**  $hi: \langle h = i \rangle$  **using** *cs-inj nreti path(1)* **by** *metis*  
**have**  $\langle l' = k \rangle$  **using** *less(2) ilh unfolding hi* **by** (*metis is-kth-obs-def*)  
**thus**  $\langle False \rangle$  **using** *lk'* **by** *simp*  
**next**  
**assume**  $\langle k < l' \wedge l < k \rangle$   
**hence**  $kl': \langle k < l' \rangle$  **and**  $lk: \langle l < k \rangle$  **by** *auto*  
**hence**  $ij': \langle i' < j' \rangle$  **and**  $ji: \langle j < i \rangle$  **using** *less(2,3) ilj ilj' kth-obs-mono* **by** *auto*  
**have**  $nreti': \langle \pi i' \neq return \rangle$  **by** (*metis csij ii'nret ij' last-cs path(2) term-path-stable less-imp-le*)  
**obtain**  $h'$  **where**  $ilh': \langle is-kth-obs \pi l h' \rangle$  **using** *all-in- $\pi'$  ilj no-kth-obs-missing-cs path(1) path(2) kl' lk'*  
*ilj' kth-obs-stable* **by** *metis*  
**hence**  $\langle cs^\pi j = cs^\pi h' \rangle$  **using** *cseq lk ilj* **by** *blast*  
**hence**  $\langle cs^\pi i' = cs^\pi h' \rangle$  **using** *csji* **by** *auto*  
**hence**  $hi: \langle h' = i' \rangle$  **using** *cs-inj nreti' path(2)* **by** *metis*  
**have**  $\langle l = k \rangle$  **using** *less(3) ilh' unfolding hi* **by** (*metis is-kth-obs-def*)  
**thus**  $\langle False \rangle$  **using** *lk* **by** *simp*  
**qed**  
**qed**  
**qed**

## 2.7 Facts about Data

**lemma** *reads-restrict1*:  $\langle \sigma \upharpoonright (reads\ n) = \sigma' \upharpoonright (reads\ n) \implies \forall x \in reads\ n. \sigma\ x = \sigma'\ x \rangle$  **by** (*metis restrict-def*)  
**lemma** *reads-restrict2*:  $\langle \forall x \in reads\ n. \sigma\ x = \sigma'\ x \implies \sigma \upharpoonright (reads\ n) = \sigma' \upharpoonright (reads\ n) \rangle$  **unfolding** *restrict-def*  
**by** *auto*  
**lemma** *reads-restrict*:  $\langle \sigma \upharpoonright (reads\ n) = \sigma' \upharpoonright (reads\ n) \rangle = \langle \forall x \in reads\ n. \sigma\ x = \sigma'\ x \rangle$  **using** *reads-restrict1*  
*reads-restrict2* **by** *metis*  
**lemma** *reads-restr-suc*:  $\langle \sigma \upharpoonright (reads\ n) = \sigma' \upharpoonright (reads\ n) \implies suc\ n\ \sigma = suc\ n\ \sigma' \rangle$  **by** (*metis reads-restrict*  
*uses-suc*)  
**lemma** *reads-restr-sem*:  $\langle \sigma \upharpoonright (reads\ n) = \sigma' \upharpoonright (reads\ n) \implies \forall v \in writes\ n. sem\ n\ \sigma\ v = sem\ n\ \sigma'\ v \rangle$  **by**

(metis reads-restrict1 uses-writes)

**lemma reads-obsp:** **assumes**  $\langle \text{path } \sigma \ k = \text{path } \sigma' \ k' \rangle \langle \sigma^k \upharpoonright (\text{reads } (\text{path } \sigma \ k)) = \sigma'^{k'} \upharpoonright (\text{reads } (\text{path } \sigma \ k)) \rangle$   
**shows**  $\langle \text{obsp } \sigma \ k = \text{obsp } \sigma' \ k' \rangle$   
**using** *assms(2) uses-att*  
**unfolding** *obsp-def assms(1) reads-restrict*  
**apply** (*cases*  $\langle \text{att } (\text{path } \sigma' \ k') \rangle$ )  
**by** *auto*

**lemma no-writes-unchanged0:** **assumes**  $\langle \forall l < k. v \notin \text{writes}(\text{path } \sigma \ l) \rangle$  **shows**  $\langle (\sigma^k) v = \sigma v \rangle$  **using** *assms*  
**proof** (*induction*  $\langle k \rangle$ )  
**case** 0 **thus**  $\langle ?\text{case} \rangle$  **by**(*auto simp add: kth-state-def*)  
**next**  
**case** (*Suc k*)  
**hence**  $\langle (\sigma^k) v = \sigma v \rangle$  **by** *auto*  
**moreover**  
**have**  $\langle \sigma^{\text{Suc } k} = \text{snd } (\text{step } (\text{path } \sigma \ k, \sigma^k)) \rangle$  **by** (*metis kth-state-suc*)  
**hence**  $\langle \sigma^{\text{Suc } k} = \text{sem } (\text{path } \sigma \ k) (\sigma^k) \rangle$  **by** (*metis step-suc-sem snd-conv*)  
**moreover**  
**have**  $\langle v \notin \text{writes } (\text{path } \sigma \ k) \rangle$  **using** *Suc.premis* **by** *blast*  
**ultimately**  
**show**  $\langle ?\text{case} \rangle$  **using** *writes* **by** *metis*  
**qed**

**lemma written-read-dd:** **assumes**  $\langle \text{is-path } \pi \rangle \langle v \in \text{reads } (\pi \ k) \rangle \langle v \in \text{writes } (\pi \ j) \rangle \langle j < k \rangle$  **obtains** *l* **where**  $\langle k \text{ dd}^{\pi, v} \rightarrow l \rangle$   
**proof** –  
**let**  $\langle ?l \rangle = \langle \text{GREATEST } l. l < k \wedge v \in \text{writes } (\pi \ l) \rangle$   
**have**  $\langle ?l < k \rangle$  **by** (*metis (no-types, lifting) GreatestI-ex-nat assms(3) assms(4) less-or-eq-imp-le*)  
**moreover**  
**have**  $\langle v \in \text{writes } (\pi \ ?l) \rangle$  **by** (*metis (no-types, lifting) GreatestI-nat assms(3) assms(4) less-or-eq-imp-le*)  
**hence**  $\langle v \in \text{reads } (\pi \ k) \cap \text{writes } (\pi \ ?l) \rangle$  **using** *assms(2)* **by** *auto*  
**moreover**  
**note** *is-ddi-def*  
**have**  $\langle \forall l \in \{?l..<k\}. v \notin \text{writes } (\pi \ l) \rangle$  **by** (*auto, metis (lifting, no-types) Greatest-le-nat le-antisym nat-less-le*)  
**ultimately**  
**have**  $\langle k \text{ dd}^{\pi, v} \rightarrow ?l \rangle$  **using** *assms(1)* **unfolding** *is-ddi-def* **by** *blast*  
**thus**  $\langle \text{thesis} \rangle$  **using** *that* **by** *simp*  
**qed**

**lemma no-writes-unchanged:** **assumes**  $\langle k \leq l \rangle \langle \forall j \in \{k..<l\}. v \notin \text{writes}(\text{path } \sigma \ j) \rangle$  **shows**  $\langle (\sigma^l) v = (\sigma^k) v \rangle$   
**using** *assms*  
**proof** (*induction*  $\langle l - k \rangle$  *arbitrary:*  $\langle l \rangle$ )  
**case** 0 **thus**  $\langle ?\text{case} \rangle$  **by**(*auto*)  
**next**  
**case** (*Suc lk l*)  
**hence** *kl*:  $\langle k < l \rangle$  **by** *auto*  
**then obtain** *l'* **where** *lsuc*:  $\langle l = \text{Suc } l' \rangle$  **using** *lessE* **by** *blast*  
**hence**  $\langle lk = l' - k \rangle$  **using** *Suc* **by** *auto*  
**moreover**  
**have**  $\langle \forall j \in \{k..<l'\}. v \notin \text{writes } (\text{path } \sigma \ j) \rangle$  **using** *Suc(4) lsuc* **by** *auto*  
**ultimately**  
**have**  $\langle (\sigma^{l'}) v = (\sigma^k) v \rangle$  **using** *Suc(1)[of*  $\langle l' \rangle$  *lsuc kl* **by** *fastforce*  
**moreover**

**have**  $\langle \sigma^l = \text{snd} ( \text{step} ( \text{path } \sigma \ l', \sigma^{l'} ) ) \rangle$  **by**  $( \text{metis } \text{kth-state-suc } \text{lsuc} )$   
**hence**  $\langle \sigma^l = \text{sem} ( \text{path } \sigma \ l' ) ( \sigma^{l'} ) \rangle$  **by**  $( \text{metis } \text{step-suc-sem } \text{snd-conv} )$   
**moreover**  
**have**  $\langle l' < l \rangle \langle k \leq l' \rangle$  **using**  $\text{kl } \text{lsuc}$  **by**  $\text{auto}$   
**hence**  $\langle v \notin \text{writes} ( \text{path } \sigma \ l' ) \rangle$  **using**  $\text{Suc.prem}(2)$  **by**  $\text{auto}$   
**ultimately**  
**show**  $\langle ?\text{case} \rangle$  **using**  $\text{writes}$  **by**  $\text{metis}$   
**qed**

**lemma ddi-value:** **assumes**  $\langle l \text{ dd} ( \text{path } \sigma ) . v \rightarrow k \rangle$  **shows**  $\langle ( \sigma^l ) v = ( \sigma^{\text{Suc } k} ) v \rangle$   
**using**  $\text{assms } \text{no-writes-unchanged} [ \text{of } \langle \text{Suc } k \rangle \langle l \rangle \langle v \rangle \langle \sigma \rangle ]$  **unfolding**  $\text{is-ddi-def}$  **by**  $\text{auto}$

**lemma written-value:** **assumes**  $\langle \text{path } \sigma \ l = \text{path } \sigma' \ l' \rangle \langle \sigma^l \upharpoonright \text{reads} ( \text{path } \sigma \ l ) = \sigma^{l'} \upharpoonright \text{reads} ( \text{path } \sigma' \ l' ) \rangle \langle v \in \text{writes} ( \text{path } \sigma \ l ) \rangle$   
**shows**  $\langle ( \sigma^{\text{Suc } l} ) v = ( \sigma'^{\text{Suc } l'} ) v \rangle$   
**by**  $( \text{metis } \text{assms } \text{reads-restr-sem } \text{snd-conv } \text{step-suc-sem } \text{kth-state-suc} )$

## 2.8 Facts about Contradicting Paths

**lemma obsp-contradict:** **assumes**  $\text{csk}: \langle \text{cs}^{\text{path } \sigma} k = \text{cs}^{\text{path } \sigma'} k' \rangle$  **and**  $\text{obs}: \langle \text{obs}_p \sigma \ k \neq \text{obs}_p \sigma' \ k' \rangle$  **shows**  $\langle ( \sigma', k' ) \text{ c } ( \sigma, k ) \rangle$   
**proof** –  
**have**  $\text{pk}: \langle \text{path } \sigma \ k = \text{path } \sigma' \ k' \rangle$  **using**  $\text{assms } \text{last-cs}$  **by**  $\text{metis}$   
**hence**  $\langle \sigma^k \upharpoonright ( \text{reads} ( \text{path } \sigma \ k ) ) \neq \sigma'^{k'} \upharpoonright ( \text{reads} ( \text{path } \sigma' \ k' ) ) \rangle$  **using**  $\text{obs } \text{reads-obs}_p [ \text{OF } \text{pk} ]$  **by**  $\text{auto}$   
**thus**  $\langle ( \sigma', k' ) \text{ c } ( \sigma, k ) \rangle$  **using**  $\text{contradicts.intros}(2) [ \text{OF } \text{csk} [ \text{symmetric} ] ]$  **by**  $\text{auto}$   
**qed**

**lemma missing-cs-contradicts:** **assumes**  $\text{notin}: \langle \neg ( \exists k'. \text{cs}^{\text{path } \sigma} k = \text{cs}^{\text{path } \sigma'} k' ) \rangle$  **and**  $\text{converge}: \langle k < n \rangle$   
 $\langle \text{cs}^{\text{path } \sigma} n = \text{cs}^{\text{path } \sigma'} n' \rangle$  **shows**  $\langle \exists j'. ( \sigma', j' ) \text{ c } ( \sigma, k ) \rangle$   
**proof** –  
**let**  $\langle ?\pi \rangle = \langle \text{path } \sigma \rangle$   
**let**  $\langle ?\pi' \rangle = \langle \text{path } \sigma' \rangle$   
**have**  $\text{init}: \langle ?\pi \ 0 = ?\pi' \ 0 \rangle$  **unfolding**  $\text{path-def}$  **by**  $\text{auto}$   
**have**  $\text{path}: \langle \text{is-path } ?\pi \rangle \langle \text{is-path } ?\pi' \rangle$  **using**  $\text{path-is-path}$  **by**  $\text{auto}$   
**obtain**  $j \ j'$  **where**  $\text{csj}: \langle \text{cs}^{?\pi} j = \text{cs}^{?\pi'} j' \rangle$  **and**  $\text{cd}: \langle k \text{ cd}^{?\pi} \rightarrow j \rangle$  **and**  $\text{suc}: \langle ?\pi ( \text{Suc } j ) \neq ?\pi' ( \text{Suc } j' ) \rangle$  **using**  $\text{converged-cd-diverge} [ \text{OF } \text{path } \text{init } \text{notin } \text{converge} ]$  .  
**have**  $\text{less}: \langle \text{cs}^{?\pi} j < \text{cs}^{?\pi} k \rangle$  **using**  $\text{cd } \text{cd-is-cs-less}$  **by**  $\text{auto}$   
**have**  $\text{nretj}: \langle ?\pi \ j \neq \text{return} \rangle$  **by**  $( \text{metis } \text{cd } \text{is-cdi-def } \text{term-path-stable } \text{less-imp-le} )$   
**have**  $\text{cs}: \langle ?\pi \ j \ \text{cs}^{?\pi'} \ j' = j \rangle$  **using**  $\text{csj } \text{cs-select-id } \text{nretj } \text{path-is-path}$  **by**  $\text{metis}$   
**have**  $\langle ( \sigma', j' ) \text{ c } ( \sigma, k ) \rangle$  **using**  $\text{contradicts.intros}(1) [ \text{of } \langle ?\pi' \rangle \langle j' \rangle \langle ?\pi \rangle \langle k \rangle \langle \sigma \rangle \langle \sigma' \rangle, \text{unfolded } \text{cs} ]$   $\text{less } \text{suc } \text{csj}$  **by**  $\text{metis}$   
**thus**  $\langle ?\text{thesis} \rangle$  **by**  $\text{blast}$   
**qed**

**theorem obs-neq-contradicts-term:** **fixes**  $\sigma \ \sigma'$  **defines**  $\pi: \langle \pi \equiv \text{path } \sigma \rangle$  **and**  $\pi': \langle \pi' \equiv \text{path } \sigma' \rangle$  **assumes**  $\text{ret}: \langle \pi \ n = \text{return} \rangle \langle \pi' \ n' = \text{return} \rangle$  **and**  $\text{obsne}: \langle \text{obs } \sigma \neq \text{obs } \sigma' \rangle$   
**shows**  $\langle \exists k \ k'. ( ( \sigma', k' ) \text{ c } ( \sigma, k ) \wedge \pi \ k \in \text{dom} ( \text{att} ) ) \vee ( ( \sigma, k ) \text{ c } ( \sigma', k' ) \wedge \pi' \ k' \in \text{dom} ( \text{att} ) ) \rangle$   
**proof** –  
**have**  $\text{path}: \langle \text{is-path } \pi \rangle \langle \text{is-path } \pi' \rangle$  **using**  $\pi \ \pi' \ \text{path-is-path}$  **by**  $\text{auto}$   
**obtain**  $k1$  **where**  $\text{neg}: \langle \text{obs } \sigma \ k1 \neq \text{obs } \sigma' \ k1 \rangle$  **using**  $\text{obsne } \text{ext} [ \text{of } \langle \text{obs } \sigma \rangle \langle \text{obs } \sigma' \rangle ]$  **by**  $\text{blast}$   
**hence**  $\langle ( \exists k \ i \ i'. \text{is-kth-obs } \pi \ k \ i \wedge \text{is-kth-obs } \pi' \ k \ i' \wedge \text{obs}_p \sigma \ i \neq \text{obs}_p \sigma' \ i' \wedge \text{cs}^\pi \ i = \text{cs}^{\pi'} \ i' ) \vee ( \exists k \ i. \text{is-kth-obs } \pi \ k \ i \wedge \neg ( \exists i'. \text{cs}^\pi \ i = \text{cs}^{\pi'} \ i' ) ) \vee ( \exists k \ i'. \text{is-kth-obs } \pi' \ k \ i' \wedge \neg ( \exists i. \text{cs}^\pi \ i = \text{cs}^{\pi'} \ i' ) ) \rangle$   
**proof**  $( \text{cases rule: } \text{option-neq-cases} )$

**case** (*none2*  $x$ )  
**have**  $\text{notin}\pi'$ :  $\langle \neg (\exists l. \text{is-kth-obs } \pi' k1 l) \rangle$  **using** *none2*(2)  $\pi'$  *obs-none-no-kth-obs* **by** *auto*  
**obtain**  $i$  **where**  $\text{in}\pi$ :  $\langle \text{is-kth-obs } \pi k1 i \rangle$  **using** *obs-some-kth-obs*[of  $\langle \sigma \rangle \langle k1 \rangle$ ] *none2*(1)  $\pi$  **by** *auto*  
**obtain**  $l j$  **where**  $\langle \text{is-kth-obs } \pi l j \rangle \langle \neg (\exists j'. \text{cs}^\pi j = \text{cs}^{\pi'} j') \rangle$  **using** *path*  $\text{in}\pi$   $\text{notin}\pi'$  **by** (*metis* *no-kth-obs-missing-cs*)  
**thus**  $\langle ?thesis \rangle$  **by** *blast*  
**next**  
**case** (*none1*  $x$ )  
**have**  $\text{notin}\pi$ :  $\langle \neg (\exists l. \text{is-kth-obs } \pi k1 l) \rangle$  **using** *none1*(1)  $\pi$  *obs-none-no-kth-obs* **by** *auto*  
**obtain**  $i'$  **where**  $\text{in}\pi'$ :  $\langle \text{is-kth-obs } \pi' k1 i' \rangle$  **using** *obs-some-kth-obs*[of  $\langle \sigma' \rangle \langle k1 \rangle$ ] *none1*(2)  $\pi'$  **by** *auto*  
**obtain**  $l j$  **where**  $\langle \text{is-kth-obs } \pi' l j \rangle \langle \neg (\exists j'. \text{cs}^\pi j' = \text{cs}^{\pi'} j) \rangle$  **using** *path*  $\text{in}\pi'$   $\text{notin}\pi$  **by** (*metis* *no-kth-obs-missing-cs*)  
**thus**  $\langle ?thesis \rangle$  **by** *blast*  
**next**  
**case** (*some*  $x y$ )  
**obtain**  $i$  **where**  $\text{in}\pi$ :  $\langle \text{is-kth-obs } \pi k1 i \rangle$  **using** *obs-some-kth-obs*[of  $\langle \sigma \rangle \langle k1 \rangle$ ] *some*  $\pi$  **by** *auto*  
**obtain**  $i'$  **where**  $\text{in}\pi'$ :  $\langle \text{is-kth-obs } \pi' k1 i' \rangle$  **using** *obs-some-kth-obs*[of  $\langle \sigma' \rangle \langle k1 \rangle$ ] *some*  $\pi'$  **by** *auto*  
**show**  $\langle ?thesis \rangle$  **proof** (*cases*)  
**assume**  $*$ :  $\langle \text{cs}^\pi i = \text{cs}^{\pi'} i' \rangle$   
**have**  $\langle \text{obsp } \sigma i = \text{obs } \sigma k1 \rangle$  **by** (*metis* *obs-def*  $\pi$   $\text{in}\pi$  *kth-obs-unique* *the-equality*)  
**moreover**  
**have**  $\langle \text{obsp } \sigma' i' = \text{obs } \sigma' k1 \rangle$  **by** (*metis* *obs-def*  $\pi'$   $\text{in}\pi'$  *kth-obs-unique* *the-equality*)  
**ultimately**  
**have**  $\langle \text{obsp } \sigma i \neq \text{obsp } \sigma' i' \rangle$  **using** *neq* **by** *auto*  
**thus**  $\langle ?thesis \rangle$  **using**  $*$   $\text{in}\pi$   $\text{in}\pi'$  **by** *blast*  
**next**  
**assume**  $*$ :  $\langle \text{cs}^\pi i \neq \text{cs}^{\pi'} i' \rangle$   
**note** *kth-obs-cs-missing-cs*[OF *path*  $\text{in}\pi$   $\text{in}\pi'$   $*$ ]  
**thus**  $\langle ?thesis \rangle$  **by** *metis*  
**qed**  
**qed**  
**thus**  $\langle ?thesis \rangle$  **proof** (*cases* *rule: three-cases*)  
**case** 1  
**then obtain**  $k i i'$  **where**  $\text{iki}$ :  $\langle \text{is-kth-obs } \pi k i \rangle \langle \text{is-kth-obs } \pi' k i' \rangle$  **and**  $\text{obsne}$ :  $\langle \text{obsp } \sigma i \neq \text{obsp } \sigma' i' \rangle$  **and**  
 $\text{csi}$ :  $\langle \text{cs}^\pi i = \text{cs}^{\pi'} i' \rangle$  **by** *auto*  
**note** *obsp-contradict*[OF  $\text{csi}$ [*unfolded*  $\pi$   $\pi'$ ]  $\text{obsne}$ ]  
**moreover**  
**have**  $\langle \pi i \in \text{dom } \text{att} \rangle$  **using**  $\text{iki}$  *unfolding* *is-kth-obs-def* **by** *auto*  
**ultimately**  
**show**  $\langle ?thesis \rangle$  **by** *blast*  
**next**  
**case** 2  
**then obtain**  $k i$  **where**  $\text{iki}$ :  $\langle \text{is-kth-obs } \pi k i \rangle$  **and**  $\text{notin}\pi'$ :  $\langle \neg (\exists i'. \text{cs}^\pi i = \text{cs}^{\pi'} i') \rangle$  **by** *auto*  
**let**  $\langle ?n \rangle = \langle \text{Suc } (\text{max } n i) \rangle$   
**have**  $\text{nn}$ :  $\langle n < ?n \rangle$  **by** *auto*  
**have**  $\text{iln}$ :  $\langle i < ?n \rangle$  **by** *auto*  
**have**  $\text{retn}$ :  $\langle \pi ?n = \text{return} \rangle$  **using** *ret* *term-path-stable* *path* **by** *auto*  
**hence**  $\langle \text{cs}^\pi ?n = \text{cs}^{\pi'} n' \rangle$  **using** *ret*(2) *cs-return* **by** *auto*  
**then obtain**  $i'$  **where**  $\langle (\sigma', i') \text{ c } (\sigma, i) \rangle$  **using** *missing-cs-contradicts*[OF  $\text{notin}\pi'$ [*unfolded*  $\pi$   $\pi'$ ]  $\text{iln}$ ]  $\pi$   $\pi'$  **by**  
*auto*  
**moreover**  
**have**  $\langle \pi i \in \text{dom } \text{att} \rangle$  **using**  $\text{iki}$  *is-kth-obs-def* **by** *auto*  
**ultimately**  
**show**  $\langle ?thesis \rangle$  **by** *blast*



next

case 3

then obtain  $k i'$  where  $iki$ :  $\langle is\text{-}kth\text{-}obs\ \pi' k i' \rangle$  and  $notin\pi'$ :  $\langle \neg (\exists i. cs^\pi i = cs^{\pi'} i') \rangle$  by auto

let  $\langle ?n \rangle = \langle Suc\ (max\ n'\ i') \rangle$

have  $nn$ :  $\langle n' < ?n \rangle$  by auto

have  $iln$ :  $\langle i' < ?n \rangle$  by auto

have  $retn$ :  $\langle \pi' ?n = return \rangle$  using *ret term-path-stable path* by auto

hence  $\langle cs^\pi n = cs^{\pi'} ?n \rangle$  using *ret(1) cs-return* by auto

then obtain  $i$  where  $\langle (\sigma, i) \mathbf{c} (\sigma', i') \rangle$  using *missing-cs-contradicts notin\pi' iln \pi \pi'* by *metis* moreover

have  $\langle \pi' i' \in dom\ att \rangle$  using *iki is-kth-obs-def* by auto

ultimately

show  $\langle ?thesis \rangle$  by *blast*

qed

qed

lemma *obs-neq-some-contradicts'*: fixes  $\sigma\ \sigma'$  defines  $\pi$ :  $\langle \pi \equiv path\ \sigma \rangle$  and  $\pi'$ :  $\langle \pi' \equiv path\ \sigma' \rangle$

assumes *obsnecs*:  $\langle obsp\ \sigma\ i \neq obsp\ \sigma'\ i' \vee cs^\pi i \neq cs^{\pi'} i' \rangle$

and *iki*:  $\langle is\text{-}kth\text{-}obs\ \pi\ k\ i \rangle$  and *iki'*:  $\langle is\text{-}kth\text{-}obs\ \pi'\ k\ i' \rangle$

shows  $\langle \exists k\ k'. ((\sigma', k') \mathbf{c} (\sigma, k) \wedge \pi k \in dom\ att) \vee ((\sigma, k) \mathbf{c} (\sigma', k') \wedge \pi' k' \in dom\ att) \rangle$

using *obsnecs iki iki' proof* (*induction*  $\langle k \rangle$  *arbitrary*:  $\langle i \rangle\ \langle i' \rangle$  *rule*: *less-induct*)

case (*less k i i'*)

note *iki* =  $\langle is\text{-}kth\text{-}obs\ \pi\ k\ i \rangle$

and *iki'* =  $\langle is\text{-}kth\text{-}obs\ \pi'\ k\ i' \rangle$

have *domi*:  $\langle \pi\ i \in dom\ att \rangle$  by (*metis is-kth-obs-def domIff iki*)

have *domi'*:  $\langle \pi'\ i' \in dom\ att \rangle$  by (*metis is-kth-obs-def domIff iki'*)

note *obsnecs* =  $\langle obsp\ \sigma\ i \neq obsp\ \sigma'\ i' \vee cs^\pi i \neq cs^{\pi'} i' \rangle$

show  $\langle ?thesis \rangle$  **proof cases**

assume *csi*:  $\langle cs^\pi i = cs^{\pi'} i' \rangle$

hence  $*$ :  $\langle obsp\ \sigma\ i \neq obsp\ \sigma'\ i' \rangle$  using *obsnecs* by auto

note *obsp-contradict*[*OF - \**] *csi domi \pi \pi'*

thus  $\langle ?thesis \rangle$  by *blast*

next

assume *ncsi*:  $\langle cs^\pi i \neq cs^{\pi'} i' \rangle$

have *path*:  $\langle is\text{-}path\ \pi \rangle\ \langle is\text{-}path\ \pi' \rangle$  using  $\pi\ \pi'$  *path-is-path* by auto

have  $\pi 0$ :  $\langle \pi\ 0 = \pi'\ 0 \rangle$  unfolding  $\pi\ \pi'$  *path-def* by auto

note *kth-obs-cs-missing-cs*[*of*  $\langle \pi \rangle\ \langle \pi' \rangle\ \langle k \rangle\ \langle i \rangle\ \langle i' \rangle$ ]  $\pi\ \pi'$  *path-is-path iki iki' ncsi*

hence  $\langle (\exists l\ j. j \leq i \wedge is\text{-}kth\text{-}obs\ \pi\ l\ j \wedge \neg (\exists j'. cs^\pi j = cs^{\pi'} j')) \vee (\exists l'\ j'. j' \leq i' \wedge is\text{-}kth\text{-}obs\ \pi'\ l'\ j' \wedge \neg (\exists j. cs^\pi j = cs^{\pi'} j')) \rangle$  by *metis*

thus  $\langle ?thesis \rangle$  **proof**

assume  $\langle \exists l\ j. j \leq i \wedge is\text{-}kth\text{-}obs\ \pi\ l\ j \wedge \neg (\exists j'. cs^\pi j = cs^{\pi'} j') \rangle$

then obtain  $l\ j$  where *ji*:  $\langle j \leq i \rangle$  and *iobs*:  $\langle is\text{-}kth\text{-}obs\ \pi\ l\ j \rangle$  and *notin*:  $\langle \neg (\exists j'. cs^\pi j = cs^{\pi'} j') \rangle$  by *blast*

have *dom*:  $\langle \pi\ j \in dom\ att \rangle$  using *iobs is-kth-obs-def* by auto

obtain  $n\ n'$  where *nj*:  $\langle n < j \rangle$  and *csn*:  $\langle cs^\pi n = cs^{\pi'} n' \rangle$  and *sucn*:  $\langle \pi (Suc\ n) \neq \pi' (Suc\ n') \rangle$  and

*cdloop*:  $\langle j\ cd^\pi \rightarrow n \vee (\forall j' > n'. j'\ cd^{\pi'} \rightarrow n') \rangle$

using *missing-cd-or-loop*[*OF path \pi 0 notin*] by *blast*

show  $\langle ?thesis \rangle$  using *cdloop proof*

assume *cdjn*:  $\langle j\ cd^\pi \rightarrow n \rangle$

hence *csnj*:  $\langle cs^{\pi'} n' < cs^\pi j \rangle$  using *csn* by (*metis cd-is-cs-less*)

have *cssel*:  $\langle \pi (Suc\ (\pi\ j\ cs^{\pi'} n')) = \pi (Suc\ n) \rangle$  using *csn* by (*metis cdjn cd-not-ret cs-select-id path(1)*)

have  $\langle (\sigma', n') \mathbf{c} (\sigma, j) \rangle$  using *csnj apply(rule contradicts.intros(1))* using *cssel \pi \pi' sucn* by auto

thus  $\langle ?thesis \rangle$  using *dom* by auto

next

assume *loop*:  $\langle \forall j' > n'. j'\ cd^{\pi'} \rightarrow n' \rangle$

**show**  $\langle ?thesis \rangle$  **proof cases**  
**assume**  $in'$ :  $\langle i' \leq n' \rangle$   
**have**  $nreti'$ :  $\langle \pi' i' \neq \text{return} \rangle$  **by** (*metis le-eq-less-or-eq lessI loop not-le path(2) ret-no-cd term-path-stable*)  
**show**  $\langle ?thesis \rangle$  **proof cases**  
**assume**  $\langle \exists \iota. cs^{\pi'} i' = cs^{\pi} \iota \rangle$   
**then obtain**  $\iota$  **where**  $cs\iota$ :  $\langle cs^{\pi} \iota = cs^{\pi'} i' \rangle$  **by** *metis*  
**have**  $\iota n$ :  $\langle \iota \leq n \rangle$  **using** *cs-order-le[OF path(2,1) cs[symmetric] csn[symmetric] nreti' in']* .  
**hence**  $\iota i$ :  $\langle \iota < i \rangle$  **using** *nj ji by auto*  
**have**  $dom\iota$ :  $\langle \pi \iota \in \text{dom att} \rangle$  **using** *domi' cs last-cs by metis*  
**obtain**  $\kappa$  **where**  $i\kappa\iota$ :  $\langle \text{is-kth-obs } \pi \kappa \iota \rangle$  **using** *domi by (metis is-kth-obs-def domIff)*  
**hence**  $\kappa k$ :  $\langle \kappa < k \rangle$  **using** *\iota iki by (metis kth-obs-le-iff)*  
**obtain**  $\iota'$  **where**  $i\kappa\iota'$ :  $\langle \text{is-kth-obs } \pi' \kappa \iota' \rangle$  **using**  $\kappa k$  *iki'* **by** (*metis kth-obs-stable*)  
**have**  $\langle \iota' < i' \rangle$  **using**  $\kappa k$  *iki' i\kappa\iota'* **by** (*metis kth-obs-le-iff*)  
**hence**  $cs\iota'$ :  $\langle cs^{\pi} \iota \neq cs^{\pi'} \iota' \rangle$  **unfolding**  $cs\iota$  **using** *cs-inj[OF path(2) nreti', of \iota']* **by** *blast*  
**thus**  $\langle ?thesis \rangle$  **using** *less(1)[OF \kappa k - i\kappa\iota\iota']* **by** *auto*  
**next**  
**assume**  $notin''$ :  $\langle \neg(\exists \iota. cs^{\pi'} i' = cs^{\pi} \iota) \rangle$   
**obtain**  $\iota \iota'$  **where**  $\iota i'$ :  $\langle \iota' < i' \rangle$  **and**  $cs\iota$ :  $\langle cs^{\pi} \iota = cs^{\pi'} \iota' \rangle$  **and**  $suc\iota$ :  $\langle \pi (Suc \iota) \neq \pi' (Suc \iota') \rangle$  **and**  
 $cdloop'$ :  $\langle i' cd^{\pi'} \rightarrow \iota' \vee (\forall j > \iota. j cd^{\pi} \rightarrow \iota) \rangle$   
**using** *missing-cd-or-loop[OF path(2,1) \pi 0[symmetric] notin'']* **by** *metis*  
**show**  $\langle ?thesis \rangle$  **using**  $cdloop'$  **proof**  
**assume**  $cdjn$ :  $\langle i' cd^{\pi'} \rightarrow \iota' \rangle$   
**hence**  $csnj$ :  $\langle cs^{\pi} \iota \prec cs^{\pi'} i' \rangle$  **using**  $cs\iota$  **by** (*metis cd-is-cs-less*)  
**have**  $cssel$ :  $\langle \pi' (Suc (\pi' i cs^{\pi} \iota)) = \pi' (Suc \iota') \rangle$  **using**  $cs\iota$  **by** (*metis cdjn cd-not-ret cs-select-id*)  
 $path(2)$   
**have**  $\langle (\sigma, \iota) c (\sigma', i') \rangle$  **using**  $csnj$  **apply**(*rule contradicts.intros(1)*) **using**  $cssel \pi \pi' suc\iota$  **by** *auto*  
**thus**  $\langle ?thesis \rangle$  **using**  $domi'$  **by** *auto*  
**next**  
**assume**  $loop'$ :  $\langle \forall j > \iota. j cd^{\pi} \rightarrow \iota \rangle$   
**have**  $\iota n'$ :  $\langle \iota' < n' \rangle$  **using**  $in' \iota i'$  **by** *auto*  
**have**  $nreti'$ :  $\langle \pi' \iota' \neq \text{return} \rangle$  **by** (*metis cs last-cs le-eq-less-or-eq lessI path(1) path(2) suc\iota term-path-stable*)  
**have**  $\langle \iota < n \rangle$  **using** *cs-order[OF path(2,1) cs[symmetric] csn[symmetric] nreti' \iota n']* .  
**hence**  $\langle \iota < i \rangle$  **using** *nj ji by auto*  
**hence**  $cdi\iota$ :  $\langle i cd^{\pi} \rightarrow \iota \rangle$  **using**  $loop'$  **by** *auto*  
**hence**  $cs\iota i$ :  $\langle cs^{\pi'} \iota' \prec cs^{\pi} i \rangle$  **using**  $cs\iota$  **by** (*metis cd-is-cs-less*)  
**have**  $cssel$ :  $\langle \pi (Suc (\pi i cs^{\pi'} \iota')) = \pi (Suc \iota) \rangle$  **using**  $cs\iota$  **by** (*metis cdi\iota cd-not-ret cs-select-id*)  
 $path(1)$   
**have**  $\langle (\sigma', \iota') c (\sigma, i) \rangle$  **using**  $cs\iota i$  **apply**(*rule contradicts.intros(1)*) **using**  $cssel \pi \pi' suc\iota$  **by** *auto*  
**thus**  $\langle ?thesis \rangle$  **using**  $domi$  **by** *auto*  
**qed**  
**qed**  
**next**  
**assume**  $\langle \neg i' \leq n' \rangle$   
**hence**  $ni'$ :  $\langle n' < i' \rangle$  **by** *simp*  
**hence**  $cdin$ :  $\langle i' cd^{\pi'} \rightarrow n' \rangle$  **using**  $loop$  **by** *auto*  
**hence**  $csni$ :  $\langle cs^{\pi} n \prec cs^{\pi'} i' \rangle$  **using**  $csn$  **by** (*metis cd-is-cs-less*)  
**have**  $cssel$ :  $\langle \pi' (Suc (\pi' i cs^{\pi} n)) = \pi' (Suc n') \rangle$  **using**  $csn$  **by** (*metis cdin cd-not-ret cs-select-id*)  
 $path(2)$   
**have**  $\langle (\sigma, n) c (\sigma', i') \rangle$  **using**  $csni$  **apply**(*rule contradicts.intros(1)*) **using**  $cssel \pi \pi' sucn$  **by** *auto*  
**thus**  $\langle ?thesis \rangle$  **using**  $domi'$  **by** *auto*  
**qed**  
**qed**  
**next**

— Symmetric case as above, indices might be messy.

**assume**  $\langle \exists l j. j \leq i' \wedge \text{is-kth-obs } \pi' l j \wedge \neg (\exists j'. \text{cs}^{\pi'} j' = \text{cs}^{\pi'} j) \rangle$   
**then obtain**  $l j$  **where**  $ji'$ :  $\langle j \leq i' \rangle$  **and**  $iobs$ :  $\langle \text{is-kth-obs } \pi' l j \rangle$  **and**  $notin$ :  $\langle \neg (\exists j'. \text{cs}^{\pi'} j = \text{cs}^{\pi'} j') \rangle$  **by**  
*metis*  
**have**  $dom$ :  $\langle \pi' j \in \text{dom att} \rangle$  **using**  $iobs$  *is-kth-obs-def* **by** *auto*  
**obtain**  $n n'$  **where**  $nj$ :  $\langle n < j \rangle$  **and**  $csn$ :  $\langle \text{cs}^{\pi'} n = \text{cs}^{\pi} n' \rangle$  **and**  $sucn$ :  $\langle \pi' (\text{Suc } n) \neq \pi (\text{Suc } n') \rangle$  **and**  
 $cdloop$ :  $\langle j \text{cd}^{\pi'} \rightarrow n \vee (\forall j' > n'. j' \text{cd}^{\pi} \rightarrow n') \rangle$   
**using** *missing-cd-or-loop*[*OF path*(2,1)  $\pi 0$ [*symmetric*] ]  $notin$  **by** *metis*  
**show**  $\langle ?thesis \rangle$  **using**  $cdloop$  **proof**  
**assume**  $cdjn$ :  $\langle j \text{cd}^{\pi'} \rightarrow n \rangle$   
**hence**  $csnj$ :  $\langle \text{cs}^{\pi} n' \prec \text{cs}^{\pi'} j \rangle$  **using**  $csn$  **by** (*metis cd-is-cs-less*)  
**have**  $cssel$ :  $\langle \pi' (\text{Suc } (\pi' j \text{cs}^{\pi} n')) = \pi' (\text{Suc } n) \rangle$  **using**  $csn$  **by** (*metis cdjn cd-not-ret cs-select-id path*(2))  
**have**  $\langle (\sigma, n') \text{c } (\sigma', j) \rangle$  **using**  $csnj$  **apply**(*rule contradicts.intros*(1)) **using**  $cssel$   $\pi' \pi$   $sucn$  **by** *auto*  
**thus**  $\langle ?thesis \rangle$  **using**  $dom$  **by** *auto*  
**next**  
**assume**  $loop$ :  $\langle \forall j' > n'. j' \text{cd}^{\pi} \rightarrow n' \rangle$   
**show**  $\langle ?thesis \rangle$  **proof** *cases*  
**assume**  $in'$ :  $\langle i \leq n' \rangle$   
**have**  $nreti$ :  $\langle \pi i \neq \text{return} \rangle$  **by** (*metis le-eq-less-or-eq lessI loop not-le path*(1) *ret-no-cd term-path-stable*)  
**show**  $\langle ?thesis \rangle$  **proof** *cases*  
**assume**  $\langle \exists \iota. \text{cs}^{\pi} i = \text{cs}^{\pi'} \iota \rangle$   
**then obtain**  $\iota$  **where**  $cs\iota$ :  $\langle \text{cs}^{\pi'} \iota = \text{cs}^{\pi} i \rangle$  **by** *metis*  
**have**  $\iota n$ :  $\langle \iota \leq n \rangle$  **using** *cs-order-le*[*OF path cs\iota*[*symmetric*]  $csn$ [*symmetric*]  $nreti$   $in'$ ] .  
**hence**  $\iota i'$ :  $\langle \iota < i' \rangle$  **using**  $nj$   $ji'$  **by** *auto*  
**have**  $dom\iota$ :  $\langle \pi' \iota \in \text{dom att} \rangle$  **using**  $dom\iota$   $cs\iota$  *last-cs* **by** *metis*  
**obtain**  $\kappa$  **where**  $i\kappa\iota$ :  $\langle \text{is-kth-obs } \pi' \kappa \iota \rangle$  **using**  $dom\iota$  **by** (*metis is-kth-obs-def domIff*)  
**hence**  $\kappa k$ :  $\langle \kappa < k \rangle$  **using**  $\iota i'$   $i\kappa\iota$  **by** (*metis kth-obs-le-iff*)  
**obtain**  $\iota'$  **where**  $i\kappa\iota'$ :  $\langle \text{is-kth-obs } \pi \kappa \iota' \rangle$  **using**  $\kappa k$   $i\kappa\iota$  **by** (*metis kth-obs-stable*)  
**have**  $\langle \iota' < i \rangle$  **using**  $\kappa k$   $i\kappa\iota$   $i\kappa\iota'$  **by** (*metis kth-obs-le-iff*)  
**hence**  $cs\iota'$ :  $\langle \text{cs}^{\pi'} \iota \neq \text{cs}^{\pi} \iota' \rangle$  **unfolding**  $cs\iota$  **using** *cs-inj*[*OF path*(1)  $nreti$ , *of*  $\langle \iota' \rangle$ ] **by** *blast*  
**thus**  $\langle ?thesis \rangle$  **using** *less*(1)[*OF*  $\kappa k - i\kappa\iota' i\kappa\iota$ ] **by** *auto*  
**next**  
**assume**  $notin''$ :  $\langle \neg (\exists \iota. \text{cs}^{\pi} i = \text{cs}^{\pi'} \iota) \rangle$   
**obtain**  $\iota \iota'$  **where**  $\iota i$ :  $\langle \iota' < i \rangle$  **and**  $cs\iota$ :  $\langle \text{cs}^{\pi'} \iota = \text{cs}^{\pi} \iota' \rangle$  **and**  $suc\iota$ :  $\langle \pi' (\text{Suc } \iota) \neq \pi (\text{Suc } \iota') \rangle$  **and**  
 $cdloop'$ :  $\langle i \text{cd}^{\pi} \rightarrow \iota' \vee (\forall j > \iota. j \text{cd}^{\pi'} \rightarrow \iota) \rangle$   
**using** *missing-cd-or-loop*[*OF path*  $\pi 0$   $notin''$ ] **by** *metis*  
**show**  $\langle ?thesis \rangle$  **using**  $cdloop'$  **proof**  
**assume**  $cdjn$ :  $\langle i \text{cd}^{\pi} \rightarrow \iota' \rangle$   
**hence**  $csnj$ :  $\langle \text{cs}^{\pi'} \iota \prec \text{cs}^{\pi} i \rangle$  **using**  $cs\iota$  **by** (*metis cd-is-cs-less*)  
**have**  $cssel$ :  $\langle \pi (\text{Suc } (\pi j \text{cs}^{\pi'} \iota)) = \pi (\text{Suc } \iota') \rangle$  **using**  $cs\iota$  **by** (*metis cdjn cd-not-ret cs-select-id*  
*path*(1))  
**have**  $\langle (\sigma', \iota) \text{c } (\sigma, i) \rangle$  **using**  $csnj$  **apply**(*rule contradicts.intros*(1)) **using**  $cssel$   $\pi' \pi$   $suc\iota$  **by** *auto*  
**thus**  $\langle ?thesis \rangle$  **using**  $dom\iota$  **by** *auto*  
**next**  
**assume**  $loop'$ :  $\langle \forall j > \iota. j \text{cd}^{\pi'} \rightarrow \iota \rangle$   
**have**  $\iota n'$ :  $\langle \iota' < n' \rangle$  **using**  $in' \iota i$  **by** *auto*  
**have**  $nreti'$ :  $\langle \pi \iota' \neq \text{return} \rangle$  **by** (*metis cs\iota last-cs le-eq-less-or-eq lessI path*(1) *path*(2)  $suc\iota$   
*term-path-stable*)  
**have**  $\langle \iota < n \rangle$  **using** *cs-order*[*OF path cs\iota*[*symmetric*]  $csn$ [*symmetric*]  $nreti' \iota n'$ ] .  
**hence**  $\langle \iota < i' \rangle$  **using**  $nj$   $ji'$  **by** *auto*  
**hence**  $cd\iota$ :  $\langle \iota' \text{cd}^{\pi'} \rightarrow \iota \rangle$  **using**  $loop'$  **by** *auto*  
**hence**  $cs\iota'$ :  $\langle \text{cs}^{\pi} \iota' \prec \text{cs}^{\pi'} i' \rangle$  **using**  $cs\iota$  **by** (*metis cd-is-cs-less*)  
**have**  $cssel$ :  $\langle \pi' (\text{Suc } (\pi' j \text{cs}^{\pi} \iota')) = \pi' (\text{Suc } \iota) \rangle$  **using**  $cs\iota$  **by** (*metis cd\iota cd-not-ret cs-select-id*

*path*(2))

**have**  $\langle (\sigma, \iota') \mathbf{c} (\sigma', i') \rangle$  **using** *cs $\iota'$*  **apply**(*rule contradicts.intros*(1)) **using** *cssel*  $\pi' \pi$  *suc $\iota$*  **by** *auto*  
**thus**  $\langle ?thesis \rangle$  **using** *dom $\iota'$*  **by** *auto*

**qed**

**qed**

**next**

**assume**  $\langle \neg i \leq n' \rangle$

**hence** *ni*:  $\langle n' < i \rangle$  **by** *simp*

**hence** *cdin*:  $\langle i \text{ cd}^\pi \rightarrow n' \rangle$  **using** *loop* **by** *auto*

**hence** *csni'*:  $\langle \text{cs}^{\pi'} n \prec \text{cs}^\pi i \rangle$  **using** *csn* **by** (*metis cd-is-cs-less*)

**have** *cssel*:  $\langle \pi (\text{Suc } (\pi \downarrow \text{cs}^{\pi'} n)) = \pi (\text{Suc } n') \rangle$  **using** *csn* **by** (*metis cdin cd-not-ret cs-select-id path*(1))

**have**  $\langle (\sigma', n) \mathbf{c} (\sigma, i) \rangle$  **using** *csni'* **apply**(*rule contradicts.intros*(1)) **using** *cssel*  $\pi' \pi$  *suc $n$*  **by** *auto*

**thus**  $\langle ?thesis \rangle$  **using** *dom $i$*  **by** *auto*

**qed**

**qed**

**qed**

**qed**

**qed**

**theorem** *obs-neq-some-contradicts*: **fixes**  $\sigma \sigma'$  **defines**  $\pi$ :  $\langle \pi \equiv \text{path } \sigma \rangle$  **and**  $\pi'$ :  $\langle \pi' \equiv \text{path } \sigma' \rangle$

**assumes** *obsne*:  $\langle \text{obs } \sigma \ k \neq \text{obs } \sigma' \ k \rangle$  **and** *not-none*:  $\langle \text{obs } \sigma \ k \neq \text{None} \rangle \langle \text{obs } \sigma' \ k \neq \text{None} \rangle$

**shows**  $\langle \exists k k'. ((\sigma', k') \mathbf{c} (\sigma, k) \wedge \pi \ k \in \text{dom } \text{att}) \vee ((\sigma, k) \mathbf{c} (\sigma', k') \wedge \pi' \ k' \in \text{dom } \text{att}) \rangle$

**proof** –

**obtain** *i* **where** *iki*:  $\langle \text{is-kth-obs } \pi \ k \ i \rangle$  **using** *not-none*(1) **by** (*metis*  $\pi$  *obs-some-kth-obs*)

**obtain** *i'* **where** *iki'*:  $\langle \text{is-kth-obs } \pi' \ k \ i' \rangle$  **using** *not-none*(2) **by** (*metis*  $\pi'$  *obs-some-kth-obs*)

**have**  $\langle \text{obs } \sigma \ i = \text{obs } \sigma' \ k \rangle$  **by** (*metis*  $\pi$  *iki* *kth-obs-unique obs-def the-equality*)

**moreover**

**have**  $\langle \text{obs } \sigma' \ i' = \text{obs } \sigma' \ k \rangle$  **by** (*metis*  $\pi'$  *iki'* *kth-obs-unique obs-def the-equality*)

**ultimately**

**have** *obs $pne$* :  $\langle \text{obs } \sigma \ i \neq \text{obs } \sigma' \ i' \rangle$  **using** *obsne* **by** *auto*

**show**  $\langle ?thesis \rangle$  **using** *obs-neq-some-contradicts'*[*OF* - *iki*[*unfolded*  $\pi$ ] *iki'*[*unfolded*  $\pi'$ ]] **using** *obs $pne$*   $\pi \pi'$  **by** *metis*

**qed**

**theorem** *obs-neq-ret-contradicts*: **fixes**  $\sigma \sigma'$  **defines**  $\pi$ :  $\langle \pi \equiv \text{path } \sigma \rangle$  **and**  $\pi'$ :  $\langle \pi' \equiv \text{path } \sigma' \rangle$

**assumes** *ret*:  $\langle \pi \ n = \text{return} \rangle$  **and** *obsne*:  $\langle \text{obs } \sigma' \ i \neq \text{obs } \sigma \ i \rangle$  **and** *obs*:  $\langle \text{obs } \sigma' \ i \neq \text{None} \rangle$

**shows**  $\langle \exists k k'. ((\sigma', k') \mathbf{c} (\sigma, k) \wedge \pi \ k \in \text{dom } (\text{att})) \vee ((\sigma, k) \mathbf{c} (\sigma', k') \wedge \pi' \ k' \in \text{dom } (\text{att})) \rangle$

**proof** (*cases*  $\langle \exists j k'. \text{is-kth-obs } \pi' \ j \ k' \wedge (\nexists k. \text{cs}^\pi \ k = \text{cs}^{\pi'} \ k') \rangle$ )

**case** *True*

**obtain** *l k'* **where** *jk'*:  $\langle \text{is-kth-obs } \pi' \ l \ k' \rangle$  **and** *unmatched*:  $\langle (\nexists k. \text{cs}^\pi \ k = \text{cs}^{\pi'} \ k') \rangle$  **using** *True* **by** *blast*

**have**  $\pi 0$ :  $\langle \pi \ 0 = \pi' \ 0 \rangle$  **using**  $\pi \pi'$  *path0* **by** *auto*

**obtain** *j j'* **where** *csj*:  $\langle \text{cs}^\pi \ j = \text{cs}^{\pi'} \ j' \rangle$  **and** *cd*:  $\langle k' \text{ cd}^{\pi'} \rightarrow j' \rangle$  **and** *suc*:  $\langle \pi (\text{Suc } j) \neq \pi' (\text{Suc } j') \rangle$

**using** *converged-cd-diverge-return*[*of*  $\langle \pi' \rangle \langle \pi \rangle \langle k' \rangle \langle n \rangle$ ] *ret unmatched path-is-path*  $\pi \pi' \pi 0$  **by** *metis*

**hence**  $*$ :  $\langle (\sigma, j) \mathbf{c} (\sigma', k') \rangle$  **using** *contradicts.intros*(1)[*of*  $\langle \pi \rangle \langle j \rangle \langle \pi' \rangle \langle k' \rangle \langle \sigma' \rangle \langle \sigma \rangle$ , *unfolded csj*]  $\pi \pi'$

**using** *cd-is-cs-less cd-not-ret cs-select-id* **by** *auto*

**have**  $\langle \pi' \ k' \in \text{dom } (\text{att}) \rangle$  **using** *jk'* **by** (*meson domIff is-kth-obs-def*)

**thus**  $\langle ?thesis \rangle$  **using**  $*$  **by** *blast*

**next**

**case** *False*

**hence**  $*$ :  $\langle \bigwedge j k'. \text{is-kth-obs } \pi' \ j \ k' \implies \exists k. \text{cs}^\pi \ k = \text{cs}^{\pi'} \ k' \rangle$  **by** *auto*

**obtain** *k'* **where** *k'*:  $\langle \text{is-kth-obs } \pi' \ i \ k' \rangle$  **using** *obs*  $\pi'$  *obs-some-kth-obs* **by** *blast*

**obtain** *l* **where**  $\langle \text{is-kth-obs } \pi \ i \ l \rangle$  **using**  $*$   $\pi \pi' \ k'$  *no-kth-obs-missing-cs path-is-path* **by** *metis*

**thus**  $\langle ?thesis \rangle$  **using**  $\pi \pi'$  *obs obs-neq-some-contradicts obs-none-no-kth-obs obsne* **by** *metis*

**qed**

## 2.9 Facts about Critical Observable Paths

**lemma contradicting-in-cp:** **assumes**  $leq: \langle \sigma =_L \sigma' \rangle$  **and**  $cseq: \langle cs^{path} \sigma k = cs^{path} \sigma' k' \rangle$   
**and**  $readv: \langle v \in reads(path \sigma k) \rangle$  **and**  $vneq: \langle (\sigma^k) v \neq (\sigma'^{k'}) v \rangle$  **shows**  $\langle ((\sigma, k), (\sigma', k')) \in cp \rangle$   
**using**  $cseq$   $readv$   $vneq$  **proof** (*induction*  $\langle k+k' \rangle$  *arbitrary:*  $\langle k \rangle \langle k' \rangle \langle v \rangle$  *rule:* *less-induct*)  
**fix**  $k k' v$   
**assume**  $csk: \langle cs^{path} \sigma k = cs^{path} \sigma' k' \rangle$   
**assume**  $vread: \langle v \in reads(path \sigma k) \rangle$   
**assume**  $vneq: \langle (\sigma^k) v \neq (\sigma'^{k'}) v \rangle$   
**assume**  $IH: \langle \bigwedge ka k'a v. ka + k'a < k + k' \implies cs^{path} \sigma ka = cs^{path} \sigma' k'a \implies v \in reads(path \sigma ka) \implies (\sigma^{ka}) v \neq (\sigma'^{k'a}) v \implies ((\sigma, ka), (\sigma', k'a)) \in cp \rangle$

**define**  $\pi$  **where**  $\langle \pi \equiv path \sigma \rangle$

**define**  $\pi'$  **where**  $\langle \pi' \equiv path \sigma' \rangle$

**have**  $path: \langle \pi = path \sigma \rangle \langle \pi' = path \sigma' \rangle$  **using**  $\pi$ -*def*  $\pi'$ -*def* *path-is-path* **by** *auto*

**have**  $ip: \langle is-path \pi \rangle \langle is-path \pi' \rangle$  **using** *path path-is-path* **by** *auto*

**have**  $\pi 0: \langle \pi' 0 = \pi 0 \rangle$  **unfolding** *path path-def* **by** *auto*

**have**  $vread': \langle v \in reads(path \sigma' k') \rangle$  **using**  $csk$   $vread$  **by** (*metis last-cs*)

**have**  $cseq: \langle cs^{\pi'} k' = cs^{\pi} k \rangle$  **using**  $csk$  *path* **by** *simp*

**show**  $\langle ((\sigma, k), (\sigma', k')) \in cp \rangle$  **proof cases**

**assume**  $vnw: \langle \forall l < k. v \notin writes(\pi l) \rangle$

**hence**  $\sigma v: \langle (\sigma^k) v = \sigma v \rangle$  **by** (*metis no-writes-unchanged0 path(1)*)

**show**  $\langle ?thesis \rangle$  **proof cases**

**assume**  $vnw': \langle \forall l < k'. v \notin writes(\pi' l) \rangle$

**hence**  $\sigma v': \langle (\sigma'^{k'}) v = \sigma' v \rangle$  **by** (*metis no-writes-unchanged0 path(2)*)

**with**  $\sigma v$   $vneq$  **have**  $\langle \sigma v \neq \sigma' v \rangle$  **by** *auto*

**hence**  $vhigh: \langle v \in hvars \rangle$  **using**  $leq$  **unfolding** *loweq-def restrict-def* **by** (*auto, metis*)

**thus**  $\langle ?thesis \rangle$  **using**  $cp.intros(1)[OF leq csk vread vneq]$   $vnw$   $vnw'$  *path* **by** *simp*

**next**

**assume**  $\langle \neg(\forall l < k'. v \notin writes(\pi' l)) \rangle$

**then obtain**  $l'$  **where**  $kddl': \langle k' dd^{\pi', v} \rightarrow l' \rangle$  **using**  $path(2)$  *path-is-path written-read-dd vread'* **by** *blast*

**hence**  $lw': \langle v \in writes(\pi' l') \rangle$  **unfolding** *is-ddi-def* **by** *auto*

**have**  $lk': \langle l' < k' \rangle$  **by** (*metis is-ddi-def kddl'*)

**have**  $nret: \langle \pi' l' \neq return \rangle$  **using**  $lw'$  *writes-return* **by** *auto*

**have**  $notin\pi: \langle \neg(\exists l. cs^{\pi'} l' = cs^{\pi} l) \rangle$  **proof**

**assume**  $\langle \exists l. cs^{\pi'} l' = cs^{\pi} l \rangle$

**then obtain**  $l$  **where**  $cs^{\pi'} l' = cs^{\pi} l$  **..**

**note**  $csl = \langle cs^{\pi'} l' = cs^{\pi} l \rangle$

**have**  $lk: \langle l < k \rangle$  **using**  $lk'$   $cseq$   $ip$   $cs$ -*order*[*of*  $\langle \pi' \rangle \langle \pi \rangle \langle l' \rangle \langle l \rangle \langle k' \rangle \langle k \rangle$ ]  $csl$   $nret$  *path* **by** *force*

**have**  $\langle v \in writes(\pi l) \rangle$  **using**  $csl$   $lw'$  *last-cs* **by** *metis*

**thus**  $\langle False \rangle$  **using**  $lk$   $vnw$  **by** *blast*

**qed**

**from** *converged-cd-diverge*[*OF ip(2,1) pi0 notinpi lk' cseq*]

**obtain**  $i i'$  **where**  $csi: \langle cs^{\pi'} i' = cs^{\pi} i \rangle$  **and**  $lcdi: \langle l' cd^{\pi'} \rightarrow i' \rangle$  **and**  $div: \langle \pi' (Suc i') \neq \pi (Suc i) \rangle$  .

**have**  $1: \langle \pi (Suc i) = suc(\pi i) (\sigma^i) \rangle$  **by** (*metis step-suc-sem fst-conv path(1) path-suc*)

**have**  $2: \langle \pi' (Suc i') = suc(\pi' i') (\sigma'^{i'}) \rangle$  **by** (*metis step-suc-sem fst-conv path(2) path-suc*)

**have**  $3: \langle \pi' i' = \pi i \rangle$  **using**  $csi$  *last-cs* **by** *metis*

**have**  $nreads: \langle \sigma^i \upharpoonright reads(\pi i) \neq \sigma'^{i'} \upharpoonright reads(\pi i) \rangle$  **by** (*metis 1 2 3 div reads-restr-suc*)

**then obtain**  $v'$  **where**  $v'$ read:  $\langle v' \in \text{reads}(\text{path } \sigma \ i) \rangle \langle (\sigma^i) \ v' \neq (\sigma^{i'}) \ v' \rangle$  **unfolding path by** (*metis reads-restrict*)

**have**  $nreti$ :  $\langle \pi' \ i' \neq \text{return} \rangle$  **by** (*metis csi div ip(1) ip(2) last-cs lessI term-path-stable less-imp-le*)  
**have**  $ik'$ :  $\langle i' < k' \rangle$  **using** *lcdi lk' is-cdi-def* **by** *auto*  
**have**  $ik$ :  $\langle i < k \rangle$  **using** *cs-order[OF ip(2,1) csi cseq nreti ik]* .

**have**  $cpi$ :  $\langle ((\sigma, i), (\sigma', i')) \in cp \rangle$  **using** *IH[of  $\langle i \rangle \langle i' \rangle$ ]*  $v'$ read *csi ik ik' path* **by** *auto*  
**hence**  $cpi'$ :  $\langle ((\sigma', i'), (\sigma, i)) \in cp \rangle$  **using** *cp.intros(4)* **by** *blast*

**have**  $nwvi$ :  $\langle \forall j' \in \{\text{LEAST } i'. \ i < i' \wedge (\exists i. \ cs^{\text{path } \sigma'} \ i = cs^{\text{path } \sigma} \ i') .. < k\}. \ v \notin \text{writes}(\text{path } \sigma \ j') \rangle$  **using** *vnw[unfolded path]*  
**by** (*metis (poly-guards-query) atLeastLessThan-iff*)

**from** *cp.intros(3)[OF cpi' kddl'[unfolded path] lcdi[unfolded path] csk[symmetric] div[unfolded path] vneq[symmetric] nwvi]*

**show**  $\langle ?thesis \rangle$  **using** *cp.intros(4)* **by** *simp*  
**qed**

**next**

**assume**  $wv$ :  $\langle \neg (\forall l < k. \ v \notin \text{writes}(\pi \ l)) \rangle$   
**then obtain**  $l$  **where**  $kddl$ :  $\langle k \ dd^{\pi, v} \rightarrow l \rangle$  **using** *path(1) path-is-path written-read-dd vread* **by** *blast*  
**hence**  $lv$ :  $\langle v \in \text{writes}(\pi \ l) \rangle$  **unfolding** *is-ddi-def* **by** *auto*  
**have**  $lk$ :  $\langle l < k \rangle$  **by** (*metis is-ddi-def kddl*)  
**have**  $nret$ :  $\langle \pi \ l \neq \text{return} \rangle$  **using** *lv writes-return* **by** *auto*  
**have**  $nwb$ :  $\langle \forall i \in \{\text{Suc } l .. < k\}. \ v \notin \text{writes}(\pi \ i) \rangle$  **using** *kddl unfolding is-ddi-def* **by** *auto*  
**have**  $\sigma vk$ :  $\langle (\sigma^k) \ v = (\sigma^{\text{Suc } l}) \ v \rangle$  **using** *kddl ddi-value path(1)* **by** *auto*

**show**  $\langle ?thesis \rangle$  **proof cases**

**assume**  $vnw'$ :  $\langle \forall l < k'. \ v \notin \text{writes}(\pi' \ l) \rangle$   
**hence**  $\sigma v'$ :  $\langle (\sigma^{k'}) \ v = \sigma' \ v \rangle$  **by** (*metis no-writes-unchanged0 path(2)*)

**have**  $\text{notin}\pi'$ :  $\langle \neg (\exists l'. \ cs^{\pi} \ l = cs^{\pi'} \ l') \rangle$  **proof**

**assume**  $\langle \exists l'. \ cs^{\pi} \ l = cs^{\pi'} \ l' \rangle$

**then obtain**  $l'$  **where**  $cs^{\pi} \ l = cs^{\pi'} \ l' ..$

**note**  $csl = \langle cs^{\pi} \ l = cs^{\pi'} \ l' \rangle$

**have**  $lk$ :  $\langle l' < k' \rangle$  **using** *lk cseq ip cs-order[of  $\langle \pi \rangle \langle \pi' \rangle \langle l \rangle \langle l' \rangle \langle k \rangle \langle k' \rangle$ ]*  $csl$   $nret$  **by** *metis*

**have**  $\langle v \in \text{writes}(\pi' \ l') \rangle$  **using** *csl lv last-cs* **by** *metis*

**thus**  $\langle \text{False} \rangle$  **using** *lk vnw'* **by** *blast*

**qed**

**from** *converged-cd-diverge[OF ip(1,2)  $\pi 0$ [symmetric] notin $\pi'$  lk cseq[symmetric]]*

**obtain**  $i \ i'$  **where**  $csi$ :  $\langle cs^{\pi'} \ i' = cs^{\pi} \ i \rangle$  **and**  $lcdi$ :  $\langle l \ cd^{\pi} \rightarrow i \rangle$  **and**  $div$ :  $\langle \pi \ (\text{Suc } i) \neq \pi' \ (\text{Suc } i') \rangle$  **by** *metis*

**have** 1:  $\langle \pi \ (\text{Suc } i) = \text{suc}(\pi \ i) \ (\sigma^i) \rangle$  **by** (*metis step-suc-sem fst-conv path(1) path-suc*)

**have** 2:  $\langle \pi' \ (\text{Suc } i') = \text{suc}(\pi' \ i') \ (\sigma^{i'}) \rangle$  **by** (*metis step-suc-sem fst-conv path(2) path-suc*)

**have** 3:  $\langle \pi' \ i' = \pi \ i \rangle$  **using** *csi last-cs* **by** *metis*

**have**  $nreads$ :  $\langle \sigma^i \upharpoonright \text{reads}(\pi \ i) \neq \sigma^{i'} \upharpoonright \text{reads}(\pi \ i) \rangle$  **by** (*metis 1 2 3 div reads-restr-suc*)

**have**  $\text{contri}$ :  $\langle (\sigma', i') \ c(\sigma, i) \rangle$  **using** *contradicts.intros(2)[OF csi path nreads]* .

**have**  $nreti$ :  $\langle \pi \ i \neq \text{return} \rangle$  **by** (*metis csi div ip(1) ip(2) last-cs lessI term-path-stable less-imp-le*)

**have**  $ik$ :  $\langle i < k \rangle$  **using** *lcdi lk is-cdi-def* **by** *auto*

**have**  $ik'$ :  $\langle i' < k' \rangle$  **using** *cs-order[OF ip(1,2) csi[symmetric] cseq[symmetric] nreti ik]* .

**have**  $nreads$ :  $\langle \sigma^i \upharpoonright reads(\pi i) \neq \sigma^{i'} \upharpoonright reads(\pi i) \rangle$  **by** (*metis 1 2 3 div reads-restr-suc*)  
**then obtain**  $v'$  **where**  $v'read$ :  $\langle v' \in reads(path \sigma i) \rangle \langle (\sigma^i) v' \neq (\sigma^{i'}) v' \rangle$  **unfolding path by** (*metis reads-restrict*)

**have**  $cp_i$ :  $\langle ((\sigma, i), (\sigma', i')) \in cp \rangle$  **using** *IH[of <i> <i'>]*  $v'read$   $csi$   $ik$   $ik'$  **path by** *auto*  
**hence**  $cp_i'$ :  $\langle ((\sigma', i'), (\sigma, i)) \in cp \rangle$  **using** *cp.intros(4)* **by** *blast*

**have**  $vnwi$ :  $\langle \forall j' \in \{LEAST i'a. i' < i'a \wedge (\exists i. cs^{path} \sigma i = cs^{path} \sigma' i'a) .. <k'\} . v \notin writes(path \sigma' j') \rangle$   
**using**  $vnw'$  [*unfolded path*]  
**by** (*metis (poly-guards-query) atLeastLessThan-iff*)

**from**  $cp.intros(3)$  [*OF cp\_i kddl[unfolded path] ldi[unfolded path] csk div[unfolded path] vneq vnwi*]

**show**  $\langle ?thesis \rangle$  **using** *cp.intros(4)* **by** *simp*  
**next**

**assume**  $\langle \neg (\forall l < k'. v \notin writes(\pi' l)) \rangle$   
**then obtain**  $l'$  **where**  $kddl'$ :  $\langle k' dd^{\pi', v} \rightarrow l' \rangle$  **using** *path(2)* *path-is-path* *written-read-dd*  $vread'$  **by** *blast*  
**hence**  $lw'$ :  $\langle v \in writes(\pi' l') \rangle$  **unfolding** *is-ddi-def* **by** *auto*  
**have**  $lk'$ :  $\langle l' < k' \rangle$  **by** (*metis is-ddi-def kddl'*)  
**have**  $nretl'$ :  $\langle \pi' l' \neq return \rangle$  **using**  $lw'$  *writes-return* **by** *auto*  
**have**  $nwb'$ :  $\langle \forall i' \in \{Suc l' .. <k'\} . v \notin writes(\pi' i') \rangle$  **using**  $kddl'$  **unfolding** *is-ddi-def* **by** *auto*  
**have**  $\sigma vk'$ :  $\langle (\sigma^{ik'}) v = (\sigma^{i' Suc l'}) v \rangle$  **using**  $kddl'$  *ddi-value* *path(2)* **by** *auto*

**show**  $\langle ?thesis \rangle$  **proof cases**

**assume**  $csl$ :  $\langle cs^\pi l = cs^{\pi'} l' \rangle$   
**hence**  $\pi l$ :  $\langle \pi l = \pi' l' \rangle$  **by** (*metis last-cs*)  
**have**  $\sigma vl$ :  $\langle (\sigma^{Suc l}) v \neq (\sigma^{i' Suc l'}) v \rangle$  **by** (*metis \sigma vk \sigma vk' vneq*)  
**have**  $r\sigma$ :  $\langle \sigma^l \upharpoonright reads(\pi l) \neq \sigma^{l'} \upharpoonright reads(\pi l) \rangle$  **using** *path*  $\pi l$   $\sigma vl$  *written-value*  $lv$  **by** *blast*  
**then obtain**  $v'$  **where**  $v'read$ :  $\langle v' \in reads(path \sigma l) \rangle \langle (\sigma^l) v' \neq (\sigma^{l'}) v' \rangle$  **unfolding path by** (*metis reads-restrict*)

**have**  $cpl$ :  $\langle ((\sigma, l), (\sigma', l')) \in cp \rangle$  **using** *IH[of <l> <l'>]*  $v'read$   $csl$   $lk$   $lk'$  **path by** *auto*  
**show**  $\langle ((\sigma, k), (\sigma', k')) \in cp \rangle$  **using**  $cp.intros(2)$  [*OF cpl kddl[unfolded path] kddl'[unfolded path] csk vneq*] .

**next**

**assume**  $csl$ :  $\langle cs^\pi l \neq cs^{\pi'} l' \rangle$   
**show**  $\langle ?thesis \rangle$  **proof cases**

**assume**  $\langle \exists i'. cs^\pi l = cs^{\pi'} i' \rangle$   
**then obtain**  $i'$  **where**  $csl_i'$ :  $\langle cs^\pi l = cs^{\pi'} i' \rangle$  **by** *blast*  
**have**  $ilne'$ :  $\langle i' \neq l' \rangle$  **using**  $csl$   $csl_i'$  **by** *auto*  
**have**  $ij'$ :  $\langle i' < k' \rangle$  **using** *cs-order* [*OF ip csl\_i' cseq[symmetric] nret lk*] .  
**have**  $iw'$ :  $\langle v \in writes(\pi' i') \rangle$  **using**  $lv$   $csl_i'$  *last-cs* **by** *metis*  
**have**  $il'$ :  $\langle i' < l' \rangle$  **using**  $kddl'$   $ilne'$   $ij'$   $iw'$  **unfolding** *is-ddi-def* **by** *auto*  
**have**  $nreti'$ :  $\langle \pi' i' \neq return \rangle$  **using**  $csl_i'$  *nret last-cs* **by** *metis*

**have**  $l'notin\pi$ :  $\langle \neg (\exists i. cs^{\pi'} l' = cs^\pi i) \rangle$  **proof**

**assume**  $\langle \exists i. cs^{\pi'} l' = cs^\pi i \rangle$   
**then obtain**  $i$  **where**  $csil$ :  $\langle cs^{\pi'} l' = cs^\pi i \rangle$  **by** *metis*  
**have**  $ik$ :  $\langle i < k \rangle$  **using** *cs-order* [*OF ip(2,1) csil[symmetric] cseq nretl' lk*] .  
**have**  $li$ :  $\langle l < i \rangle$  **using** *cs-order* [*OF ip(2,1) csl\_i'[symmetric] csil[symmetric] nreti' il*] .  
**have**  $iv$ :  $\langle v \in writes(\pi i) \rangle$  **using**  $lv$   $csil$  *last-cs* **by** *metis*  
**show**  $\langle False \rangle$  **using**  $kddl$   $ik$   $li$   $iv$  *is-ddi-def* **by** *auto*

qed

**obtain**  $n\ n'$  **where**  $csn: \langle cs^\pi n = cs^{\pi'} n' \rangle$  **and**  $lcdn': \langle l' cd^{\pi'} \rightarrow n' \rangle$  **and**  $sucn: \langle \pi (Suc\ n) \neq \pi' (Suc\ n') \rangle$  **and**  $in': \langle i' \leq n' \rangle$

**using** *converged-cd-diverge-cs* [*OF ip(2,1) csli'[symmetric] il' l'notin $\pi$  lk' cseq*] **by** *metis*

— Can apply the IH to  $n$  and  $n'$

**have** 1:  $\langle \pi (Suc\ n) = suc\ (\pi\ n)\ (\sigma^n) \rangle$  **by** (*metis step-suc-sem fst-conv path(1) path-suc*)

**have** 2:  $\langle \pi' (Suc\ n') = suc\ (\pi'\ n')\ (\sigma^{m'}) \rangle$  **by** (*metis step-suc-sem fst-conv path(2) path-suc*)

**have** 3:  $\langle \pi'\ n' = \pi\ n \rangle$  **using** *csn last-cs* **by** *metis*

**have**  $nreads: \langle \sigma^n \upharpoonright reads\ (\pi\ n) \neq \sigma^{m'} \upharpoonright reads\ (\pi\ n) \rangle$  **by** (*metis 1 2 3 sucn reads-restr-suc*)

**then obtain**  $v'$  **where**  $v'read: \langle v' \in reads\ (path\ \sigma\ n) \rangle$   $\langle (\sigma^n)\ v' \neq (\sigma^{m'})\ v' \rangle$  **by** (*metis path(1) reads-restrict*)

**moreover**

**have**  $nl': \langle n' < l' \rangle$  **using** *lcdn' is-cdi-def* **by** *auto*

**have**  $nk': \langle n' < k' \rangle$  **using**  $nl'\ lk'$  **by** *simp*

**have**  $nretl': \langle \pi'\ n' \neq return \rangle$  **by** (*metis ip(2) nl' nretl' term-path-stable less-imp-le*)

**have**  $nk: \langle n < k \rangle$  **using** *cs-order*[*OF ip(2,1) csn[symmetric] cseq nretl' nk'*] .

**hence**  $lenn: \langle n+n' < k+k' \rangle$  **using**  $nk'$  **by** *auto*

**ultimately**

**have**  $\langle ((\sigma, n), (\sigma', n')) \in cp \rangle$  **using** *IH csn path* **by** *auto*

**hence**  $ncp: \langle ((\sigma', n'), (\sigma, n)) \in cp \rangle$  **using** *cp.intros(4)* **by** *auto*

**have**  $nles: \langle n < (LEAST\ i'.\ n < i' \wedge (\exists i. cs^{\pi'} i = cs^\pi i')) \rangle$  (**is**  $\langle - < (LEAST\ i.\ ?P\ i) \rangle$ ) **using**  $nk\ cseq$  *LeastI*[*of*  $\langle ?P \rangle \langle k \rangle$ ] **by** *metis*

**moreover**

**have**  $ln: \langle l \leq n \rangle$  **using** *cs-order-le*[*OF ip(2,1) csli'[symmetric] csn[symmetric] nreti' in'*] .

**ultimately**

**have**  $lles: \langle Suc\ l \leq (LEAST\ i'.\ n < i' \wedge (\exists i. cs^{\pi'} i = cs^\pi i')) \rangle$  **by** *auto*

**have**  $nwcseq: \langle \forall j' \in \{LEAST\ i'.\ n < i' \wedge (\exists i. cs^{\pi'} i = cs^\pi i')..<k\}. v \notin writes\ (\pi\ j') \rangle$  **proof**

**fix**  $j'$  **assume**  $*$ :  $\langle j' \in \{LEAST\ i'.\ n < i' \wedge (\exists i. cs^{\pi'} i = cs^\pi i')..<k\} \rangle$

**hence**  $\langle (LEAST\ i'.\ n < i' \wedge (\exists i. cs^{\pi'} i = cs^\pi i')) \leq j' \rangle$  **by** (*metis (poly-guards-query) atLeast-LessThan-iff*)

**hence**  $\langle Suc\ l \leq j' \rangle$  **using**  $lles$  **by** *auto*

**moreover**

**have**  $\langle j' < k \rangle$  **using**  $*$  **by** (*metis (poly-guards-query) atLeastLessThan-iff*)

**ultimately have**  $\langle j' \in \{Suc\ l..<k\} \rangle$  **by** (*metis (poly-guards-query) atLeastLessThan-iff*)

**thus**  $\langle v \notin writes\ (\pi\ j') \rangle$  **using**  $nwb$  **by** *auto*

qed

**from** *cp.intros(3)*[*OF ncp, folded path, OF kddl' lcdn' cseq sucn[symmetric] vneq[symmetric] nwcseq*]

**have**  $\langle ((\sigma', k'), (\sigma, k)) \in cp \rangle$  .

**thus**  $\langle ((\sigma, k), (\sigma', k')) \in cp \rangle$  **using** *cp.intros(4)* **by** *auto*

**next**

**assume**  $lnotin\pi': \langle \neg (\exists i'. cs^\pi l = cs^{\pi'} i') \rangle$

**show**  $\langle ?thesis \rangle$  **proof cases**

**assume**  $\langle \exists i. cs^\pi i = cs^{\pi'} l' \rangle$

**then obtain**  $i$  **where**  $csli: \langle cs^\pi i = cs^{\pi'} l' \rangle$  **by** *blast*

**have**  $ilne: \langle i \neq l \rangle$  **using**  $csli\ csli$  **by** *auto*

**have**  $ij: \langle i < k \rangle$  **using** *cs-order*[*OF ip(2,1) csli[symmetric] cseq nretl' lk'*] .

**have**  $iv: \langle v \in writes\ (\pi\ i) \rangle$  **using**  $lv'\ csli\ last-cs$  **by** *metis*

**have**  $il: \langle i < l \rangle$  **using**  $kddl\ ilne\ ij\ iv$  **unfolding** *is-ddi-def* **by** *auto*

**have**  $nreti: \langle \pi\ i \neq return \rangle$  **using**  $csli\ nretl'\ last-cs$  **by** *metis*



**obtain**  $n\ n'$  **where**  $csn: \langle cs^\pi n = cs^{\pi'} n' \rangle$  **and**  $lcdn: \langle l\ cd^\pi \rightarrow n \rangle$  **and**  $sucn: \langle \pi (Suc\ n) \neq \pi' (Suc\ n') \rangle$  **and**  $ilen: \langle i \leq n \rangle$   
**using** *converged-cd-diverge-cs* [*OF ip csli il lnotin $\pi'$  lk cseq[symmetric]*] **by** *metis*

— Can apply the IH to  $n$  and  $n'$

**have** 1:  $\langle \pi (Suc\ n) = suc\ (\pi\ n)\ (\sigma^n) \rangle$  **by** (*metis step-suc-sem fst-conv path(1) path-suc*)

**have** 2:  $\langle \pi' (Suc\ n') = suc\ (\pi'\ n')\ (\sigma^{m'}) \rangle$  **by** (*metis step-suc-sem fst-conv path(2) path-suc*)

**have** 3:  $\langle \pi'\ n' = \pi\ n \rangle$  **using** *csn last-cs* **by** *metis*

**have**  $nreads: \langle \sigma^n \upharpoonright reads\ (\pi\ n) \neq \sigma^{m'} \upharpoonright reads\ (\pi\ n) \rangle$  **by** (*metis 1 2 3 sucn reads-restr-suc*)

**then obtain**  $v'$  **where**  $v'read: \langle v' \in reads\ (path\ \sigma\ n) \rangle \langle (\sigma^n)\ v' \neq (\sigma^{m'})\ v' \rangle$  **by** (*metis path(1) reads-restrict*)

**moreover**

**have**  $nl: \langle n < l \rangle$  **using** *lcdn is-cdi-def* **by** *auto*

**have**  $nk: \langle n < k \rangle$  **using** *nl lk* **by** *simp*

**have**  $nretn: \langle \pi\ n \neq return \rangle$  **by** (*metis ip(1) nl nret term-path-stable less-imp-le*)

**have**  $nk': \langle n' < k' \rangle$  **using** *cs-order* [*OF ip csn cseq[symmetric] nretn nk*] .

**hence**  $lenn: \langle n+n' < k+k' \rangle$  **using** *nk* **by** *auto*

**ultimately**

**have**  $nep: \langle ((\sigma, n), (\sigma', n')) \in ep \rangle$  **using** *IH csn path* **by** *auto*

**have**  $nles': \langle n' < (LEAST\ i'.\ n' < i' \wedge (\exists i. cs^\pi i = cs^{\pi'} i')) \rangle$  (**is**  $\langle - < (LEAST\ i. ?P\ i) \rangle$ ) **using**  $nk'$  *cseq LeastI* [*of*  $\langle ?P \rangle \langle k' \rangle$ ] **by** *metis*

**moreover**

**have**  $ln': \langle l' \leq n' \rangle$  **using** *cs-order-le* [*OF ip csli csn nreti ilen*] .

**ultimately**

**have**  $lles': \langle Suc\ l' \leq (LEAST\ i'.\ n' < i' \wedge (\exists i. cs^\pi i = cs^{\pi'} i')) \rangle$  **by** *auto*

**have**  $nwcseq': \langle \forall j' \in \{(LEAST\ i'.\ n' < i' \wedge (\exists i. cs^\pi i = cs^{\pi'} i'))..<k'\}. v \notin writes\ (\pi'\ j') \rangle$  **proof**

**fix**  $j'$  **assume**  $*$ :  $\langle j' \in \{(LEAST\ i'.\ n' < i' \wedge (\exists i. cs^\pi i = cs^{\pi'} i'))..<k'\} \rangle$

**hence**  $\langle (LEAST\ i'.\ n' < i' \wedge (\exists i. cs^\pi i = cs^{\pi'} i')) \leq j' \rangle$  **by** (*metis (poly-guards-query) atLeastLessThan-iff*)

**hence**  $\langle Suc\ l' \leq j' \rangle$  **using**  $lles'$  **by** *auto*

**moreover**

**have**  $\langle j' < k' \rangle$  **using**  $*$  **by** (*metis (poly-guards-query) atLeastLessThan-iff*)

**ultimately have**  $\langle j' \in \{Suc\ l'..<k'\} \rangle$  **by** (*metis (poly-guards-query) atLeastLessThan-iff*)

**thus**  $\langle v \notin writes\ (\pi'\ j') \rangle$  **using**  $nwb'$  **by** *auto*

**qed**

**from** *cp.intros(3)* [*OF nep, folded path, OF kddl lcdn cseq[symmetric] sucn vneq nwcseq*]

**show**  $\langle ((\sigma, k), (\sigma', k')) \in ep \rangle$  .

**next**

**assume**  $l'notin\pi: \langle \neg (\exists i. cs^\pi i = cs^{\pi'} l') \rangle$

**define**  $m$  **where**  $\langle m \equiv 0::nat \rangle$

**define**  $m'$  **where**  $\langle m' \equiv 0::nat \rangle$

**have**  $esm: \langle cs^\pi m = cs^{\pi'} m' \rangle$  **unfolding**  $m\text{-def}\ m'\text{-def}\ cs\ 0$  **by** (*metis  $\pi 0$* )

**have**  $ml: \langle m < l \vee m' < l' \rangle$  **using**  $esm\ csl$  **unfolding**  $m\text{-def}\ m'\text{-def}$  **by** (*metis neq0-conv*)

**have**  $\langle \exists n\ n'. cs^\pi n = cs^{\pi'} n' \wedge \pi (Suc\ n) \neq \pi' (Suc\ n') \wedge$

$(l\ cd^\pi \rightarrow n \wedge (\forall j' \in \{(LEAST\ i'.\ n' < i' \wedge (\exists i. cs^\pi i = cs^{\pi'} i'))..<k'\}. v \notin writes\ (\pi'\ j'))$

$\vee l'\ cd^{\pi'} \rightarrow n' \wedge (\forall j \in \{(LEAST\ i. n < i \wedge (\exists i'. cs^{\pi'} i' = cs^\pi i))..<k'\}. v \notin writes\ (\pi\ j)) \rangle$

**using**  $esm\ ml$  **proof** (*induction*  $\langle k+k'-(m+m') \rangle$  *arbitrary:*  $\langle m \rangle \langle m' \rangle$  *rule: less-induct*)

**case** (*less m m'*)

**note**  $csm = \langle cs^\pi m = cs^{\pi'} m' \rangle$   
**note**  $lm = \langle m < l \vee m' < l' \rangle$   
**note**  $IH = \langle \bigwedge n n' .$   
 $k + k' - (n + n') < k + k' - (m + m') \implies$   
 $cs^\pi n = cs^{\pi'} n' \implies$   
 $n < l \vee n' < l' \implies ?thesis \rangle$   
**show**  $\langle ?thesis \rangle$  **using**  $lm$  **proof**  
**assume**  $ml: \langle m < l \rangle$   
**obtain**  $n n'$  **where**  $mn: \langle m \leq n \rangle$  **and**  $csn: \langle cs^\pi n = cs^{\pi'} n' \rangle$  **and**  $lcdn: \langle l \text{ cd}^\pi \rightarrow n \rangle$  **and**  $suc: \langle \pi$   
 $(Suc\ n) \neq \pi' (Suc\ n') \rangle$   
**using**  $converged-cd-diverge-cs[OF\ ip\ csm\ ml\ lnotin\ \pi' \text{ lk}\ cseq[symmetric]]$  .  
**have**  $nl: \langle n < l \rangle$  **using**  $lcdn$   $is-cdi-def$  **by**  $auto$   
**hence**  $nk: \langle n < k \rangle$  **using**  $lk$  **by**  $auto$   
**have**  $nretn: \langle \pi\ n \neq return \rangle$  **using**  $lcdn$  **by**  $(metis\ cd-not-ret)$   
**have**  $nk': \langle n' < k' \rangle$  **using**  $cs-order[OF\ ip\ csn\ cseq[symmetric]]\ nretn\ nk$  .  
**show**  $\langle ?thesis \rangle$  **proof cases**  
**assume**  $\langle \forall j' \in \{ (LEAST\ i' . n' < i' \wedge (\exists i . cs^\pi i = cs^{\pi'} i')) .. < k' \} . v \notin writes\ (\pi' j') \rangle$   
**thus**  $\langle ?thesis \rangle$  **using**  $lcdn\ csn\ suc$  **by**  $blast$   
**next**  
**assume**  $\langle \neg (\forall j' \in \{ (LEAST\ i' . n' < i' \wedge (\exists i . cs^\pi i = cs^{\pi'} i')) .. < k' \} . v \notin writes\ (\pi' j')) \rangle$   
**then obtain**  $j'$  **where**  $jin': \langle j' \in \{ (LEAST\ i' . n' < i' \wedge (\exists i . cs^\pi i = cs^{\pi'} i')) .. < k' \} \rangle$  **and**  $vwrite:$   
 $\langle v \in writes\ (\pi' j') \rangle$  **by**  $blast$   
**define**  $i'$  **where**  $\langle i' \equiv LEAST\ i' . n' < i' \wedge (\exists i . cs^\pi i = cs^{\pi'} i') \rangle$   
**have**  $Pk': \langle n' < k' \wedge (\exists k . cs^\pi k = cs^{\pi'} k') \rangle$  **(is**  $\langle ?P\ k' \rangle$ ) **using**  $nk'$   $cseq[symmetric]$  **by**  $blast$   
**have**  $ni': \langle n' < i' \rangle$  **using**  $LeastI[of\ \langle ?P \rangle, OF\ Pk']\ i'-def$  **by**  $auto$   
**obtain**  $i$  **where**  $csi: \langle cs^\pi i = cs^{\pi'} i' \rangle$  **using**  $LeastI[of\ \langle ?P \rangle, OF\ Pk']\ i'-def$  **by**  $blast$   
**have**  $ij': \langle i' \leq j' \rangle$  **using**  $jin'$   $[folded\ i'-def]$  **by**  $auto$   
**have**  $jk': \langle j' < k' \rangle$  **using**  $jin'$   $[folded\ i'-def]$  **by**  $auto$   
**have**  $jl': \langle j' \leq l' \rangle$  **using**  $kddl'\ jk'\ vwrite$  **unfolding**  $is-ddi-def$  **by**  $auto$   
**have**  $nretn': \langle \pi' n' \neq return \rangle$  **using**  $nretn\ csn\ last-cs$  **by**  $metis$   
**have**  $iln: \langle n < i \rangle$  **using**  $cs-order[OF\ ip(2,1)\ csn[symmetric]\ csi[symmetric]\ nretn'\ ni']$  .  
**hence**  $mi: \langle m < i \rangle$  **using**  $mn$  **by**  $auto$   
**have**  $nretm: \langle \pi\ m \neq return \rangle$  **by**  $(metis\ ip(1)\ mn\ nretn\ term-path-stable)$   
**have**  $mi': \langle m' < i' \rangle$  **using**  $cs-order[OF\ ip\ csm\ csi\ nretm\ mi]$  .  
**have**  $ik': \langle i' < k' \rangle$  **using**  $ij'\ jk'$  **by**  $auto$   
**have**  $nreti': \langle \pi' i' \neq return \rangle$  **by**  $(metis\ ij'\ jl'\ nretl'\ ip(2)\ term-path-stable)$   
**have**  $ik: \langle i < k \rangle$  **using**  $cs-order[OF\ ip(2,1)\ csi[symmetric]\ cseq\ nreti'\ ik']$  .  
**show**  $\langle ?thesis \rangle$  **proof cases**  
**assume**  $il: \langle i < l \rangle$   
**have**  $le: \langle k + k' - (i + i') < k + k' - (m + m') \rangle$  **using**  $mi\ mi'\ ik\ ik'$  **by**  $auto$   
**show**  $\langle ?thesis \rangle$  **using**  $IH[OF\ le]$  **using**  $csi\ il$  **by**  $blast$   
**next**  
**assume**  $\langle \neg i < l \rangle$   
**hence**  $li: \langle l \leq i \rangle$  **by**  $auto$   
**have**  $\langle i' \leq l' \rangle$  **using**  $ij'\ jl'$  **by**  $auto$   
**hence**  $il': \langle i' < l' \rangle$  **using**  $csi\ l'notin\ \pi$  **by**  $fastforce$   
**obtain**  $n n'$  **where**  $in': \langle i' \leq n' \rangle$  **and**  $csn: \langle cs^\pi n = cs^{\pi'} n' \rangle$  **and**  $lcdn': \langle l' \text{ cd}^{\pi'} \rightarrow n' \rangle$  **and**  
 $suc: \langle \pi (Suc\ n) \neq \pi' (Suc\ n') \rangle$   
**using**  $converged-cd-diverge-cs[OF\ ip(2,1)\ csi[symmetric]\ il' - lk' cseq]\ l'notin\ \pi$  **by**  $metis$   
**have**  $nk': \langle n' < k' \rangle$  **using**  $lcdn'$   $is-cdi-def\ lk'$  **by**  $auto$   
**have**  $nretn': \langle \pi' n' \neq return \rangle$  **by**  $(metis\ cd-not-ret\ lcdn')$   
**have**  $nk: \langle n < k \rangle$  **using**  $cs-order[OF\ ip(2,1)\ csn[symmetric]\ cseq\ nretn'\ nk']$  .  
**define**  $j$  **where**  $\langle j \equiv LEAST\ j . n < j \wedge (\exists j' . cs^{\pi'} j' = cs^\pi j) \rangle$   
**have**  $Pk: \langle n < k \wedge (\exists j' . cs^{\pi'} j' = cs^\pi k) \rangle$  **(is**  $\langle ?P\ k \rangle$ ) **using**  $nk\ cseq$  **by**  $blast$

**have**  $nj$ :  $\langle n < j \rangle$  **using** *LeastI*[of  $\langle ?P \rangle$ , *OF Pk*] *j-def* **by** *auto*  
**have**  $ilen$ :  $\langle i \leq n \rangle$  **using** *cs-order-le*[*OF ip(2,1)*] *csi[symmetric]* *csn[symmetric]* *nreti' in'* .  
**hence**  $lj$ :  $\langle l < j \rangle$  **using**  $li$   $nj$  **by** *simp*  
**have**  $\langle \forall l \in \{l..<k\}. v \notin \text{writes}(\pi l) \rangle$  **using** *kddl unfolding is-ddi-def* **by** *simp*  
**hence**  $nw$ :  $\langle \forall l \in \{j..<k\}. v \notin \text{writes}(\pi l) \rangle$  **using**  $lj$  **by** *auto*  
**show**  $\langle ?thesis \rangle$  **using** *csn lcdn' suc nw[unfolded j-def]* **by** *blast*  
**qed**  
**qed**  
**next**  
**assume**  $ml'$ :  $\langle m' < l' \rangle$   
**obtain**  $n n'$  **where**  $mn'$ :  $\langle m' \leq n' \rangle$  **and**  $csn$ :  $\langle cs^\pi n = cs^{\pi'} n' \rangle$  **and**  $lcdn'$ :  $\langle l' cd^{\pi'} \rightarrow n' \rangle$  **and**  
 $suc$ :  $\langle \pi (Suc\ n) \neq \pi' (Suc\ n') \rangle$   
**using** *converged-cd-diverge-cs*[*OF ip(2,1)*] *csn[symmetric]*  $ml' - lk'$  *cseq*] *l'notin $\pi$*  **by** *metis*  
**have**  $nl'$ :  $\langle n' < l' \rangle$  **using** *lcdn' is-cdi-def* **by** *auto*  
**hence**  $nk'$ :  $\langle n' < k' \rangle$  **using**  $lk'$  **by** *auto*  
**have**  $nretn'$ :  $\langle \pi' n' \neq \text{return} \rangle$  **using** *lcdn' by (metis cd-not-ret)*  
**have**  $nk$ :  $\langle n < k \rangle$  **using** *cs-order*[*OF ip(2,1)*] *csn[symmetric]* *cseq nretn' nk'* .  
**show**  $\langle ?thesis \rangle$  **proof cases**  
**assume**  $\langle \forall j \in \{(LEAST\ i.\ n < i \wedge (\exists i'.\ cs^{\pi'} i' = cs^\pi i))..<k\}. v \notin \text{writes}(\pi j) \rangle$   
**thus**  $\langle ?thesis \rangle$  **using** *lcdn' csn suc* **by** *blast*  
**next**  
**assume**  $\langle \neg(\forall j \in \{(LEAST\ i.\ n < i \wedge (\exists i'.\ cs^{\pi'} i' = cs^\pi i))..<k\}. v \notin \text{writes}(\pi j)) \rangle$   
**then obtain**  $j$  **where**  $jin$ :  $\langle j \in \{(LEAST\ i.\ n < i \wedge (\exists i'.\ cs^{\pi'} i' = cs^\pi i))..<k\} \rangle$  **and**  $vwrite$ :  
 $\langle v \in \text{writes}(\pi j) \rangle$  **by** *blast*  
**define**  $i$  **where**  $\langle i \equiv LEAST\ i.\ n < i \wedge (\exists i'.\ cs^{\pi'} i' = cs^\pi i) \rangle$   
**have**  $Pk$ :  $\langle n < k \wedge (\exists k'.\ cs^{\pi'} k' = cs^\pi k) \rangle$  **(is**  $\langle ?P\ k \rangle$ ) **using**  $nk$  *cseq* **by** *blast*  
**have**  $ni$ :  $\langle n < i \rangle$  **using** *LeastI*[of  $\langle ?P \rangle$ , *OF Pk*] *i-def* **by** *auto*  
**obtain**  $i'$  **where**  $csi$ :  $\langle cs^\pi i = cs^{\pi'} i' \rangle$  **using** *LeastI*[of  $\langle ?P \rangle$ , *OF Pk*] *i-def* **by** *metis*  
**have**  $ij$ :  $\langle i \leq j \rangle$  **using**  $jin$ [*folded i-def*] **by** *auto*  
**have**  $jk$ :  $\langle j < k \rangle$  **using**  $jin$ [*folded i-def*] **by** *auto*  
**have**  $jl$ :  $\langle j \leq l \rangle$  **using** *kddl jk vwrite unfolding is-ddi-def* **by** *auto*  
**have**  $nretn$ :  $\langle \pi n \neq \text{return} \rangle$  **using**  $nretn'$  *csn last-cs* **by** *metis*  
**have**  $iln'$ :  $\langle n' < i' \rangle$  **using** *cs-order*[*OF ip csn csi nretn ni*] .  
**hence**  $mi'$ :  $\langle m' < i' \rangle$  **using**  $mn'$  **by** *auto*  
**have**  $nretm'$ :  $\langle \pi' m' \neq \text{return} \rangle$  **by** *(metis ip(2) mn' nretn' term-path-stable)*  
**have**  $mi$ :  $\langle m < i \rangle$  **using** *cs-order*[*OF ip(2,1)*] *csm[symmetric]* *csi[symmetric]* *nretm' mi'* .  
**have**  $ik$ :  $\langle i < k \rangle$  **using**  $ij$   $jk$  **by** *auto*  
**have**  $nreti$ :  $\langle \pi i \neq \text{return} \rangle$  **by** *(metis ij ip(1) jl nret term-path-stable)*  
**have**  $ik'$ :  $\langle i' < k' \rangle$  **using** *cs-order*[*OF ip csi cseq[symmetric] nreti ik*] .  
**show**  $\langle ?thesis \rangle$  **proof cases**  
**assume**  $il'$ :  $\langle i' < l' \rangle$   
**have**  $le$ :  $\langle k + k' - (i + i') < k + k' - (m + m') \rangle$  **using**  $mi$   $mi'$   $ik$   $ik'$  **by** *auto*  
**show**  $\langle ?thesis \rangle$  **using** *IH*[*OF le*] **using** *csi il'* **by** *blast*  
**next**  
**assume**  $\langle \neg i' < l' \rangle$   
**hence**  $li'$ :  $\langle l' \leq i' \rangle$  **by** *auto*  
**have**  $\langle i \leq l \rangle$  **using**  $ij$   $jl$  **by** *auto*  
**hence**  $il$ :  $\langle i < l \rangle$  **using** *csi lnotin $\pi'$*  **by** *fastforce*  
**obtain**  $n n'$  **where**  $ilen$ :  $\langle i \leq n \rangle$  **and**  $csn$ :  $\langle cs^\pi n = cs^{\pi'} n' \rangle$  **and**  $lcdn$ :  $\langle l cd^\pi \rightarrow n \rangle$  **and**  $suc$ :  
 $\langle \pi (Suc\ n) \neq \pi' (Suc\ n') \rangle$   
**using** *converged-cd-diverge-cs*[*OF ip csi il - lk cseq[symmetric]*] *lnotin $\pi'$*  **by** *metis*  
**have**  $nk$ :  $\langle n < k \rangle$  **using** *lcdn is-cdi-def lk* **by** *auto*  
**have**  $nretn$ :  $\langle \pi n \neq \text{return} \rangle$  **by** *(metis cd-not-ret lcdn)*  
**have**  $nk'$ :  $\langle n' < k' \rangle$  **using** *cs-order*[*OF ip csn cseq[symmetric] nretn nk*] .  
**define**  $j'$  **where**  $\langle j' \equiv LEAST\ j'.\ n' < j' \wedge (\exists j.\ cs^\pi j = cs^{\pi'} j') \rangle$

**have**  $Pk'$ :  $\langle n' < k' \wedge (\exists j. cs^\pi j = cs^{\pi'} k') \rangle$  (**is**  $\langle ?P k' \rangle$ ) **using**  $nk'$  *cseq[symmetric]* **by** *blast*  
**have**  $nj'$ :  $\langle n' < j' \rangle$  **using** *LeastI*[of  $\langle ?P \rangle$ , *OF Pk'*] *j'-def* **by** *auto*  
**have**  $in'$ :  $\langle i' \leq n' \rangle$  **using** *cs-order-le*[*OF ip csi csn nreti ilen*] .  
**hence**  $lj'$ :  $\langle l' < j' \rangle$  **using**  $li'$   $nj'$  **by** *simp*  
**have**  $\langle \forall l \in \{l' .. < k'\}. v \notin \text{writes}(\pi' l) \rangle$  **using** *kddl'* **unfolding** *is-ddi-def* **by** *simp*  
**hence**  $nw'$ :  $\langle \forall l \in \{j' .. < k'\}. v \notin \text{writes}(\pi' l) \rangle$  **using**  $lj'$  **by** *auto*  
**show**  $\langle ?thesis \rangle$  **using** *csn lcdn suc nw'*[*unfolded j'-def*] **by** *blast*  
**qed**  
**qed**  
**qed**  
**qed**  
**then obtain**  $n n'$  **where**  $csn$ :  $\langle cs^\pi n = cs^{\pi'} n' \rangle$  **and**  $suc$ :  $\langle \pi (Suc n) \neq \pi' (Suc n') \rangle$   
**and**  $cdor$ :  
 $\langle l cd^\pi \rightarrow n \wedge (\forall j' \in \{LEAST i'. n' < i' \wedge (\exists i. cs^\pi i = cs^{\pi'} i') .. < k'\}. v \notin \text{writes}(\pi' j'))$   
 $\vee l' cd^{\pi'} \rightarrow n' \wedge (\forall j \in \{LEAST i. n < i \wedge (\exists i'. cs^{\pi'} i' = cs^\pi i) .. < k'\}. v \notin \text{writes}(\pi j)) \rangle$   
**by** *blast*  
**show**  $\langle ?thesis \rangle$  **using** *cdor proof*  
**assume**  $*$ :  $\langle l cd^\pi \rightarrow n \wedge (\forall j' \in \{LEAST i'. n' < i' \wedge (\exists i. cs^\pi i = cs^{\pi'} i') .. < k'\}. v \notin \text{local.writes}$   
 $(\pi' j')) \rangle$   
**hence**  $lcdn$ :  $\langle l cd^\pi \rightarrow n \rangle$  **by** *blast*  
**have**  $nowrite$ :  $\langle \forall j' \in \{LEAST i'. n' < i' \wedge (\exists i. cs^\pi i = cs^{\pi'} i') .. < k'\}. v \notin \text{local.writes}(\pi' j') \rangle$  **using**  
 $*$  **by** *blast*  
**show**  $\langle ?thesis \rangle$  **proof** (*rule cp.intros(3)*[of  $\langle \sigma \rangle \langle n \rangle \langle \sigma' \rangle \langle n' \rangle$ , *folded path*])  
**show**  $\langle l cd^\pi \rightarrow n \rangle$  **using**  $lcdn$  .  
**show**  $\langle k dd^{\pi, v} \rightarrow l \rangle$  **using** *kddl* .  
**show**  $\langle cs^\pi k = cs^{\pi'} k' \rangle$  **using** *cseq* **by** *simp*  
**show**  $\langle \pi (Suc n) \neq \pi' (Suc n') \rangle$  **using**  $suc$  **by** *simp*  
**show**  $\langle \forall j' \in \{LEAST i'. n' < i' \wedge (\exists i. cs^\pi i = cs^{\pi'} i') .. < k'\}. v \notin \text{local.writes}(\pi' j') \rangle$  **using**  
 $nowrite$  .  
**show**  $\langle (\sigma^k) v \neq (\sigma'^{k'}) v \rangle$  **using** *vneq* .  
**have**  $nk$ :  $\langle n < k \rangle$  **using**  $lcdn lk$  *is-cdi-def* **by** *auto*  
**have**  $nretn$ :  $\langle \pi n \neq \text{return} \rangle$  **using** *cd-not-ret lcdn* **by** *metis*  
**have**  $nk'$ :  $\langle n' < k' \rangle$  **using** *cs-order*[*OF ip csn cseq[symmetric] nretn nk*] .  
**hence**  $le$ :  $\langle n + n' < k + k' \rangle$  **using**  $nk$  **by** *auto*  
**moreover**  
**have** 1:  $\langle \pi (Suc n) = suc(\pi n) (\sigma^n) \rangle$  **by** (*metis step-suc-sem fst-conv path(1) path-suc*)  
**have** 2:  $\langle \pi' (Suc n') = suc(\pi' n') (\sigma^{m'}) \rangle$  **by** (*metis step-suc-sem fst-conv path(2) path-suc*)  
**have** 3:  $\langle \pi' n' = \pi n \rangle$  **using** *csn last-cs* **by** *metis*  
**have**  $nreads$ :  $\langle \sigma^n \upharpoonright \text{reads}(\pi n) \neq \sigma^{m'} \upharpoonright \text{reads}(\pi n) \rangle$  **by** (*metis 1 2 3 suc reads-restr-suc*)  
**then obtain**  $v'$  **where**  $v'$  *read*:  $\langle v' \in \text{reads}(\text{path } \sigma n) \wedge (\sigma^n) v' \neq (\sigma^{m'}) v' \rangle$  **by** (*metis path(1)*  
 $\text{reads-restrict}$ )  
**ultimately**  
**show**  $\langle ((\sigma, n), (\sigma', n')) \in cp \rangle$  **using** *IH csn path* **by** *auto*  
**qed**  
**next**  
**assume**  $*$ :  $\langle l' cd^{\pi'} \rightarrow n' \wedge (\forall j \in \{LEAST i. n < i \wedge (\exists i'. cs^{\pi'} i' = cs^\pi i) .. < k'\}. v \notin \text{writes}(\pi j)) \rangle$   
**hence**  $lcdn'$ :  $\langle l' cd^{\pi'} \rightarrow n' \rangle$  **by** *blast*  
**have**  $nowrite$ :  $\langle \forall j \in \{LEAST i. n < i \wedge (\exists i'. cs^{\pi'} i' = cs^\pi i) .. < k'\}. v \notin \text{writes}(\pi j) \rangle$  **using**  $*$  **by**  
 $blast$   
**show**  $\langle ?thesis \rangle$  **proof** (*rule cp.intros(4)*, *rule cp.intros(3)*[of  $\langle \sigma' \rangle \langle n' \rangle \langle \sigma \rangle \langle n \rangle$ , *folded path*])  
**show**  $\langle l' cd^{\pi'} \rightarrow n' \rangle$  **using**  $lcdn'$  .  
**show**  $\langle k' dd^{\pi', v} \rightarrow l' \rangle$  **using** *kddl'* .  
**show**  $\langle cs^{\pi'} k' = cs^\pi k \rangle$  **using** *cseq* .

**show**  $\langle \pi' (Suc\ n') \neq \pi (Suc\ n) \rangle$  **using** *suc* **by** *simp*  
**show**  $\langle \forall j \in \{(LEAST\ i.\ n < i \wedge (\exists i'.\ cs^{\pi'}\ i' = cs^{\pi}\ i))..<k\}.\ v \notin writes\ (\pi\ j) \rangle$  **using** *nowrite* .  
**show**  $\langle (\sigma^{k'})\ v \neq (\sigma^k)\ v \rangle$  **using** *vneq* **by** *simp*  
**have**  $nk': \langle n' < k' \rangle$  **using** *lcdn' lk' is-cdi-def* **by** *auto*  
**have**  $nretn': \langle \pi'\ n' \neq return \rangle$  **using** *cd-not-ret lcdn'* **by** *metis*  
**have**  $nk: \langle n < k \rangle$  **using** *cs-order[OF ip(2,1) csn[symmetric] cseq nretn' nk']* .  
**hence**  $le: \langle n + n' < k + k' \rangle$  **using** *nk'* **by** *auto*  
**moreover**  
**have** 1:  $\langle \pi (Suc\ n) = suc\ (\pi\ n)\ (\sigma^n) \rangle$  **by** (*metis step-suc-sem fst-conv path(1) path-suc*)  
**have** 2:  $\langle \pi' (Suc\ n') = suc\ (\pi'\ n')\ (\sigma^{m'}) \rangle$  **by** (*metis step-suc-sem fst-conv path(2) path-suc*)  
**have** 3:  $\langle \pi'\ n' = \pi\ n \rangle$  **using** *csn last-cs* **by** *metis*  
**have**  $nreads: \langle \sigma^n \upharpoonright reads\ (\pi\ n) \neq \sigma^{m'} \upharpoonright reads\ (\pi\ n) \rangle$  **by** (*metis 1 2 3 suc reads-restr-suc*)  
**then obtain**  $v'$  **where**  $v'read: \langle v' \in reads\ (path\ \sigma\ n) \rangle$   $\langle (\sigma^n)\ v' \neq (\sigma^{m'})\ v' \rangle$  **by** (*metis path(1) reads-restrict*)  
**ultimately**  
**have**  $\langle ((\sigma, n), (\sigma', n')) \in cp \rangle$  **using** *IH csn path* **by** *auto*  
**thus**  $\langle ((\sigma', n'), \sigma, n) \in cp \rangle$  **using** *cp.intros(4)* **by** *simp*  
**qed**  
**qed**  
**qed**  
**qed**  
**qed**  
**qed**  
**qed**  
**qed**

**theorem contradicting-in-cop: assumes**  $\langle \sigma =_L \sigma' \rangle$  **and**  $\langle (\sigma', k')\ c\ (\sigma, k) \rangle$  **and**  $\langle path\ \sigma\ k \in dom\ att \rangle$   
**shows**  $\langle ((\sigma, k), \sigma', k') \in cop \rangle$  **using** *assms(2)* **proof**(*cases*)  
**case** (1  $\pi'\ \pi$ )  
**define**  $j$  **where**  $\langle j \equiv \pi \upharpoonright cs^{\pi'}\ k' \rangle$   
**have**  $csj: \langle cs^{\pi}\ j = cs^{\pi'}\ k' \rangle$  **unfolding** *j-def* **using** 1 **by** (*metis cs-not-nil cs-select-is-cs(1) path-is-path*)  
**have**  $suc: \langle \pi (Suc\ j) \neq \pi' (Suc\ k') \rangle$  **using** 1 *j-def* **by** *simp*  
**have**  $kcdj: \langle k\ cd^{\pi} \rightarrow j \rangle$  **by** (*metis cs-not-nil cs-select-is-cs(2) 1(1,2) j-def path-is-path*)  
**obtain**  $v$  **where**  $readv: \langle v \in reads\ (path\ \sigma\ j) \rangle$  **and**  $vneq: \langle (\sigma^j)\ v \neq (\sigma^{k'})\ v \rangle$  **using** *suc csj unfolding 1* **by** (*metis IFC-def.suc-def 1(2) 1(3) last-cs path-suc reads-restr-suc reads-restrict*)  
**have**  $\langle ((\sigma, j), \sigma', k') \in cp \rangle$  **apply** (*rule contradicting-in-cp[OF assms(1)]*) **using** *readv vneq csj 1* **by** *auto*  
**thus**  $\langle ((\sigma, k), \sigma', k') \in cop \rangle$  **using** *kcdj suc assms(3) cop.intros(2) unfolding 1* **by** *auto*  
**next**  
**case** (2  $\pi'\ \pi$ )  
**obtain**  $v$  **where**  $readv: \langle v \in reads\ (path\ \sigma\ k) \rangle$  **and**  $vneq: \langle (\sigma^k)\ v \neq (\sigma^{k'})\ v \rangle$  **using** 2(2-4) **by** (*metis reads-restrict*)  
**have**  $\langle ((\sigma, k), \sigma', k') \in cp \rangle$  **apply** (*rule contradicting-in-cp[OF assms(1)]*) **using** *readv vneq 2* **by** *auto*  
**thus**  $\langle ((\sigma, k), \sigma', k') \in cop \rangle$  **using** *assms(3) cop.intros(1) unfolding 2* **by** *auto*  
**qed**

**theorem cop-correct-term: fixes**  $\sigma\ \sigma'$  **defines**  $\pi: \langle \pi \equiv path\ \sigma \rangle$  **and**  $\pi': \langle \pi' \equiv path\ \sigma' \rangle$   
**assumes**  $ret: \langle \pi\ n = return \rangle$   $\langle \pi'\ n' = return \rangle$  **and**  $obsne: \langle obs\ \sigma \neq obs\ \sigma' \rangle$  **and**  $leq: \langle \sigma =_L \sigma' \rangle$   
**shows**  $\langle \exists k\ k'. ((\sigma, k), \sigma', k') \in cop \vee ((\sigma', k'), \sigma, k) \in cop \rangle$   
**proof** –  
**have** \*:  $\langle \exists k\ k'. ((\sigma', k')\ c\ (\sigma, k) \wedge \pi\ k \in dom\ (att)) \vee ((\sigma, k)\ c\ (\sigma', k') \wedge \pi'\ k' \in dom\ (att)) \rangle$  **using** *obs-neq-contradicts-term ret obsne  $\pi\ \pi'$*  **by** *auto*  
**have**  $leq': \langle \sigma' =_L \sigma \rangle$  **using** *leq unfolding loweq-def* **by** *auto*

**from** \* *contradicting-in-cop*[*OF leq*] *contradicting-in-cop*[*OF leq'*] **show**  $\langle ?thesis \rangle$  **unfolding**  $\pi \pi'$  **by** *metis* **qed**

**theorem** *cop-correct-ret*: **fixes**  $\sigma \sigma'$  **defines**  $\pi$ :  $\langle \pi \equiv \text{path } \sigma \rangle$  **and**  $\pi'$ :  $\langle \pi' \equiv \text{path } \sigma' \rangle$   
**assumes** *ret*:  $\langle \pi n = \text{return} \rangle$  **and** *obsne*:  $\langle \text{obs } \sigma i \neq \text{obs } \sigma' i \rangle$  **and** *obs*:  $\langle \text{obs } \sigma' i \neq \text{None} \rangle$  **and** *leq*:  $\langle \sigma =_L \sigma' \rangle$   
**shows**  $\langle \exists k k'. ((\sigma, k), \sigma', k') \in \text{cop} \vee ((\sigma', k'), \sigma, k) \in \text{cop} \rangle$

**proof** –

**have** \*:  $\langle \exists k k'. ((\sigma', k') \text{ c } (\sigma, k) \wedge \pi k \in \text{dom } (\text{att})) \vee ((\sigma, k) \text{ c } (\sigma', k') \wedge \pi' k' \in \text{dom } (\text{att})) \rangle$

**by** (*metis* (*no-types*, *lifting*)  $\pi \pi'$  *obs obs-neq-ret-contradicts obsne ret*)

**have** *leq'*:  $\langle \sigma' =_L \sigma \rangle$  **using** *leq* **unfolding** *loweq-def* **by** *auto*

**from** \* *contradicting-in-cop*[*OF leq*] *contradicting-in-cop*[*OF leq'*] **show**  $\langle ?thesis \rangle$  **unfolding**  $\pi \pi'$  **by** *metis* **qed**

**theorem** *cop-correct-nterm*: **assumes** *obsne*:  $\langle \text{obs } \sigma k \neq \text{obs } \sigma' k \rangle$   $\langle \text{obs } \sigma k \neq \text{None} \rangle$   $\langle \text{obs } \sigma' k \neq \text{None} \rangle$   
**and** *leq*:  $\langle \sigma =_L \sigma' \rangle$

**shows**  $\langle \exists k k'. ((\sigma, k), \sigma', k') \in \text{cop} \vee ((\sigma', k'), \sigma, k) \in \text{cop} \rangle$

**proof** –

**obtain**  $k k'$  **where**  $\langle ((\sigma', k') \text{ c } (\sigma, k) \wedge \text{path } \sigma k \in \text{dom } \text{att}) \vee ((\sigma, k) \text{ c } (\sigma', k') \wedge \text{path } \sigma' k' \in \text{dom } \text{att}) \rangle$

**using** *obs-neq-some-contradicts*[*OF obsne*] **by** *metis*

**thus**  $\langle ?thesis \rangle$  **proof**

**assume** \*:  $\langle (\sigma', k') \text{ c } (\sigma, k) \wedge \text{path } \sigma k \in \text{dom } \text{att} \rangle$

**hence**  $\langle ((\sigma, k), \sigma', k') \in \text{cop} \rangle$  **using** *leq* **by** (*metis* *contradicting-in-cop*)

**thus**  $\langle ?thesis \rangle$  **using** \* **by** *blast*

**next**

**assume** \*:  $\langle (\sigma, k) \text{ c } (\sigma', k') \wedge \text{path } \sigma' k' \in \text{dom } \text{att} \rangle$

**hence**  $\langle ((\sigma', k'), \sigma, k) \in \text{cop} \rangle$  **using** *leq* **by** (*metis* *contradicting-in-cop* *loweq-def*)

**thus**  $\langle ?thesis \rangle$  **using** \* **by** *blast*

**qed**

**qed**

## 2.10 Correctness of the Characterisation

The following is our main correctness result. If there exist no critical observable paths, then the program is secure.

**theorem** *cop-correct*: **assumes**  $\langle \text{cop} = \text{empty} \rangle$  **shows**  $\langle \text{secure} \rangle$  **proof** (*rule ccontr*)

**assume**  $\langle \neg \text{secure} \rangle$

**then obtain**  $\sigma \sigma'$  **where** *leq*:  $\langle \sigma =_L \sigma' \rangle$

**and** \*\*:  $\langle \neg \text{obs } \sigma \approx \text{obs } \sigma' \vee (\text{terminates } \sigma \wedge \neg \text{obs } \sigma' \lesssim \text{obs } \sigma) \rangle$

**unfolding** *secure-def* **by** *blast*

**show**  $\langle \text{False} \rangle$  **using** \*\* **proof**

**assume**  $\langle \neg \text{obs } \sigma \approx \text{obs } \sigma' \rangle$

**then obtain**  $k$  **where**  $\langle \text{obs } \sigma k \neq \text{obs } \sigma' k \wedge \text{obs } \sigma k \neq \text{None} \wedge \text{obs } \sigma' k \neq \text{None} \rangle$

**unfolding** *obs-comp-def* *obs-prefix-def*

**by** (*metis* *kth-obs-stable* *linorder-neqE-nat* *obs-none-no-kth-obs* *obs-some-kth-obs*)

**thus**  $\langle \text{False} \rangle$  **using** *cop-correct-nterm* *leq assms* **by** *auto*

**next**

**assume** \*:  $\langle \text{terminates } \sigma \wedge \neg \text{obs } \sigma' \lesssim \text{obs } \sigma \rangle$

**then obtain**  $n$  **where** *ret*:  $\langle \text{path } \sigma n = \text{return} \rangle$

**unfolding** *terminates-def* **by** *auto*

**obtain**  $k$  **where**  $\langle \text{obs } \sigma k \neq \text{obs } \sigma' k \wedge \text{obs } \sigma' k \neq \text{None} \rangle$  **using** \* **unfolding** *obs-prefix-def* **by** *metis*

**thus**  $\langle \text{False} \rangle$  **using** *cop-correct-ret* *ret leq assms* **by** (*metis* *empty-iff*)

**qed**

**qed**

Our characterisation is not only correct, it is also precise in the way that *cp* characterises exactly the matching indices in executions for low equivalent input states where diverging data is read. This follows easily as the inverse implication to lemma *contradicting-in-cp* can be shown by simple induction.

**theorem** *cp-iff-reads-contradict*:  $\langle((\sigma,k),(\sigma',k')) \in cp \longleftrightarrow \sigma =_L \sigma' \wedge cs^{path} \sigma k = cs^{path} \sigma' k' \wedge (\exists v \in reads(path \sigma k). (\sigma^k) v \neq (\sigma'^k) v)\rangle$

**proof**

**assume**  $\langle\sigma =_L \sigma' \wedge cs^{path} \sigma k = cs^{path} \sigma' k' \wedge (\exists v \in reads(path \sigma k). (\sigma^k) v \neq (\sigma'^k) v)\rangle$

**thus**  $\langle((\sigma, k), \sigma', k') \in cp\rangle$  **using** *contradicting-in-cp* **by** *blast*

**next**

**assume**  $\langle((\sigma, k), \sigma', k') \in cp\rangle$

**thus**  $\langle\sigma =_L \sigma' \wedge cs^{path} \sigma k = cs^{path} \sigma' k' \wedge (\exists v \in reads(path \sigma k). (\sigma^k) v \neq (\sigma'^k) v)\rangle$

**proof** (*induction*)

**case** (1  $\sigma \sigma' n n' h$ )

**then show**  $\langle?case\rangle$  **by** *blast*

**next**

**case** (2  $\sigma k \sigma' k' n v n'$ )

**have**  $\langle v \in reads(path \sigma n)\rangle$  **using** 2(2) **unfolding** *is-ddi-def* **by** *auto*

**then show**  $\langle?case\rangle$  **using** 2 **by** *auto*

**next**

**case** (3  $\sigma k \sigma' k' n v l n'$ )

**have**  $\langle v \in reads(path \sigma n)\rangle$  **using** 3(2) **unfolding** *is-ddi-def* **by** *auto*

**then show**  $\langle?case\rangle$  **using** 3(4,6,8) **by** *auto*

**next**

**case** (4  $\sigma k \sigma' k'$ )

**hence**  $\langle cs^{path} \sigma k = cs^{path} \sigma' k'\rangle$  **by** *simp*

**hence**  $\langle path \sigma' k' = path \sigma k\rangle$  **by** (*metis last-cs*)

**moreover have**  $\langle\sigma' =_L \sigma\rangle$  **using** 4(2) **unfolding** *loweq-def* **by** *simp*

**ultimately show**  $\langle?case\rangle$  **using** 4 **by** *metis*

**qed**

**qed**

In the same way the inverse implication to *contradicting-in-cop* follows easily such that we obtain the following characterisation of *cop*.

**theorem** *cop-iff-contradicting*:  $\langle((\sigma,k),(\sigma',k')) \in cop \longleftrightarrow \sigma =_L \sigma' \wedge (\sigma',k') \mathfrak{c}(\sigma,k) \wedge path \sigma k \in dom\ att\rangle$

**proof**

**assume**  $\langle\sigma =_L \sigma' \wedge (\sigma', k') \mathfrak{c}(\sigma, k) \wedge path \sigma k \in dom\ att\rangle$  **thus**  $\langle((\sigma,k),(\sigma',k')) \in cop\rangle$  **using** *contradicting-in-cop* **by** *simp*

**next**

**assume**  $\langle((\sigma,k),(\sigma',k')) \in cop\rangle$

**thus**  $\langle\sigma =_L \sigma' \wedge (\sigma',k') \mathfrak{c}(\sigma,k) \wedge path \sigma k \in dom\ att\rangle$  **proof** (*cases rule: cop.cases*)

**case** 1

**then show**  $\langle?thesis\rangle$  **using** *cp-iff-reads-contradict* *contradicts.simps* **by** (*metis (full-types) reads-restrict1*)

**next**

**case** (2  $k$ )

**then show**  $\langle?thesis\rangle$  **using** *cp-iff-reads-contradict* *contradicts.simps*

**by** (*metis cd-is-cs-less cd-not-ret contradicts.intros(1) cs-select-id path-is-path*)

**qed**

**qed**

## 2.11 Correctness of the Single Path Approximation

**theorem** *cp-in-scp*: **assumes**  $\langle((\sigma,k),(\sigma',k')) \in cp\rangle$  **shows**  $\langle path \sigma k \in scp \wedge (path \sigma',k') \in scp\rangle$  **using** *assms* **proof** (*induction*  $\langle\sigma\rangle \langle k\rangle \langle\sigma'\rangle \langle k'\rangle$  *rule:cp.induct[case-names read-high dd dcd sym]*)

**case** (*read-high*  $\sigma \sigma' k k' h$ )

**have**  $\langle \sigma h = (\sigma^k) h \rangle$  **using** *read-high(5)* **by** (*simp add: no-writes-unchanged0*)  
**moreover have**  $\langle \sigma' h = (\sigma'^k) h \rangle$  **using** *read-high(6)* **by** (*simp add: no-writes-unchanged0*)  
**ultimately have**  $\langle \sigma h \neq \sigma' h \rangle$  **using** *read-high(4)* **by** *simp*  
**hence**  $*$ :  $\langle h \in hvars \rangle$  **using** *read-high(1)* **unfolding** *loweq-def* **by** (*metis Compl-iff IFC-def.restrict-def*)  
**have**  $1$ :  $\langle (\text{path } \sigma, k) \in scp \rangle$  **using** *scp.intros(1)* *read-high(3,5)*  $*$  **by** *auto*  
**have**  $\langle \text{path } \sigma k = \text{path } \sigma' k' \rangle$  **using** *read-high(2)* **by** (*metis last-cs*)  
**hence**  $\langle (\text{path } \sigma', k') \in scp \rangle$  **using** *scp.intros(1)* *read-high(3,6)*  $*$  **by** *auto*  
**thus**  $\langle ?case \rangle$  **using**  $1$  **by** *auto*  
**next**  
**case** *dd* **show**  $\langle ?case \rangle$  **using** *scp.intros(3)* *dd* **by** *auto*  
**next**  
**case** *sym* **thus**  $\langle ?case \rangle$  **by** *blast*  
**next**  
**case** (*dcd*  $\sigma k \sigma' k' n v l n'$ )  
**note** *scp.intros(4)* *is-dcdi-via-def* *cd-cs-swap* *cs-ipd*  
**have**  $1$ :  $\langle (\text{path } \sigma, n) \in scp \rangle$  **using** *dcd.IH* *dcd.hyps(2)* *dcd.hyps(3)* *scp.intros(2)* *scp.intros(3)* **by** *blast*  
**have** *csk*:  $\langle cs^{\text{path}} \sigma k = cs^{\text{path}} \sigma' k' \rangle$  **using** *cp-eq-cs[OF dcd(1)]* .  
**have** *kn*:  $\langle k < n \rangle$  **and** *kl*:  $\langle k < l \rangle$  **and** *ln*:  $\langle l < n \rangle$  **using** *dcd(2,3)* **unfolding** *is-ddi-def* *is-cdi-def* **by** *auto*  
**have** *nret*:  $\langle \text{path } \sigma k \neq \text{return} \rangle$  **using** *cd-not-ret* *dcd.hyps(3)* **by** *auto*  
**have**  $\langle k' < n' \rangle$  **using** *kn* *csk* *dcd(4)* *cs-order* *nret* *path-is-path* *last-cs* **by** *blast*  
**have**  $2$ :  $\langle (\text{path } \sigma', n') \in scp \rangle$  **proof** *cases*  
**assume** *j'ex*:  $\langle \exists j' \in \{k'..<n'\}. v \in \text{writes}(\text{path } \sigma' j') \rangle$   
**hence**  $\langle \exists j'. j' \in \{k'..<n'\} \wedge v \in \text{writes}(\text{path } \sigma' j') \rangle$  **by** *auto*  
**note**  $*$  = *GreatestI-ex-nat[OF this]*  
**define** *j'* **where**  $\langle j' == \text{GREATEST } j'. j' \in \{k'..<n'\} \wedge v \in \text{writes}(\text{path } \sigma' j') \rangle$   
**note**  $**$  =  $*$ [*of j', folded j'-def*]  
**have**  $\langle k' \leq j' \rangle$   $\langle j' < n' \rangle$  **and** *j'write*:  $\langle v \in \text{writes}(\text{path } \sigma' j') \rangle$   
**using**  $*$  *atLeastLessThan-iff j'-def* *nat-less-le* **by** *auto*  
**have** *nowrite*:  $\langle \forall i' \in \{j'..<n'\}. v \notin \text{writes}(\text{path } \sigma' i') \rangle$  **proof** (*rule, rule ccontr*)  
**fix** *i'* **assume**  $\langle i' \in \{j'..<n'\} \rangle$   $\langle \neg v \notin \text{local.writes}(\text{path } \sigma' i') \rangle$   
**hence**  $\langle i' \in \{k'..<n'\} \wedge v \in \text{local.writes}(\text{path } \sigma' i') \rangle$  **using**  $\langle k' \leq j' \rangle$  **by** *auto*  
**hence**  $\langle i' \leq j' \rangle$  **using** *Greatest-le-nat*  
**by** (*metis (no-types, lifting) atLeastLessThan-iff j'-def* *nat-less-le*)  
**thus**  $\langle \text{False} \rangle$  **using**  $\langle i' \in \{j'..<n'\} \rangle$  **by** *auto*  
**qed**  
**have**  $\langle \text{path } \sigma' n' = \text{path } \sigma n \rangle$  **using** *dcd(4)* *last-cs* **by** *metis*  
**hence**  $\langle v \in \text{reads}(\text{path } \sigma' n') \rangle$  **using** *dcd(2)* **unfolding** *is-ddi-def* **by** *auto*  
**hence** *nddj'*:  $\langle n' \text{ dd}^{\text{path}} \sigma', v \rightarrow j' \rangle$  **using** *dcd(2)* **unfolding** *is-ddi-def* **using** *nowrite*  $\langle j' < n' \rangle$  *j'write* **by**  
*auto*  
**show**  $\langle ?thesis \rangle$  **proof** *cases*  
**assume**  $\langle j' \text{ cd}^{\text{path}} \sigma' \rightarrow k' \rangle$   
**thus**  $\langle (\text{path } \sigma', n') \in scp \rangle$  **using** *scp.intros(2)* *scp.intros(3)* *dcd.IH* *nddj'* **by** *fast*  
**next**  
**assume** *jcdk'*:  $\langle \neg j' \text{ cd}^{\text{path}} \sigma' \rightarrow k' \rangle$   
**show**  $\langle ?thesis \rangle$  **proof** *cases*  
**assume**  $\langle j' = k' \rangle$   
**thus**  $\langle ?thesis \rangle$  **using** *scp.intros(3)* *dcd.IH* *nddj'* **by** *fastforce*  
**next**  
**assume**  $\langle j' \neq k' \rangle$  **hence**  $\langle k' < j' \rangle$  **using**  $\langle k' \leq j' \rangle$  **by** *auto*  
**have**  $\langle \text{path } \sigma' j' \neq \text{return} \rangle$  **using** *j'write* *writes-return* **by** *auto*  
**hence** *ipdex'*:  $\langle \exists j. j \in \{k'..j'\} \wedge \text{path } \sigma' j = \text{ipd}(\text{path } \sigma' k') \rangle$  **using** *path-is-path*  $\langle k' < j' \rangle$  *jcdk'* *is-cdi-def*  
**by** *blast*  
**define** *i'* **where**  $\langle i' == \text{LEAST } j. j \in \{k'..j'\} \wedge \text{path } \sigma' j = \text{ipd}(\text{path } \sigma' k') \rangle$   
**have** *i'pd'*:  $\langle i' \in \{k'..j'\} \rangle$   $\langle \text{path } \sigma' i' = \text{ipd}(\text{path } \sigma' k') \rangle$  **unfolding** *i'-def* **using** *LeastI-ex[OF ipdex']* **by**  
*simp-all*



```

have *:  $\langle \forall i \in \{k'..<i'\}. \text{path } \sigma' i \neq \text{ipd } (\text{path } \sigma' k') \rangle$  proof (rule, rule ccontr)
  fix i assume *:  $\langle i \in \{k'..<i'\} \rangle \langle \neg \text{path } \sigma' i \neq \text{ipd } (\text{path } \sigma' k') \rangle$ 
  hence **:  $\langle i \in \{k'..j'\} \wedge \text{path } \sigma' i = \text{ipd } (\text{path } \sigma' k') \rangle$  (is  $\langle ?P i \rangle$ ) using iipd'(1) by auto
  thus  $\langle \text{False} \rangle$  using Least-le[of  $\langle ?P \rangle \langle i \rangle$ ] i'-def * by auto
qed
have  $\langle i' \neq k' \rangle$  using iipd'(2) by (metis csk last-cs nret path-in-nodes ipd-not-self)
hence  $\langle k' < i' \rangle$  using iipd'(1) by simp
hence csi':  $\langle \text{cs}^{\text{path}} \sigma' i' = [n \leftarrow \text{cs}^{\text{path}} \sigma' k' . \text{ipd } n \neq \text{path } \sigma' i'] @ [\text{path } \sigma' i'] \rangle$  using cs-ipd[OF iipd'(2)]
*] by fast

have ncdk':  $\langle \neg n' \text{cd}^{\text{path}} \sigma' \rightarrow k' \rangle$  using  $\langle j' < n' \rangle \langle k' < j' \rangle$  cdi-prefix jcdk' less-imp-le-nat by blast
hence ncdk:  $\langle \neg n \text{cd}^{\text{path}} \sigma \rightarrow k \rangle$  using cd-cs-swap csk dcd(4) by blast
have ipdex:  $\langle \exists i. i \in \{k..n\} \wedge \text{path } \sigma i = \text{ipd } (\text{path } \sigma k) \rangle$  (is  $\langle \exists i. ?P i \rangle$ ) proof cases
  assume *:  $\langle \text{path } \sigma n = \text{return} \rangle$ 
  from path-ret-ipd[of  $\langle \text{path } \sigma \rangle \langle k \rangle \langle n \rangle$ , OF path-is-path nret *]
  obtain i where  $\langle ?P i \rangle$  by fastforce thus  $\langle ?thesis \rangle$  by auto
next
  assume *:  $\langle \text{path } \sigma n \neq \text{return} \rangle$ 
  show  $\langle ?thesis \rangle$  using not-cd-impl-ipd [of  $\langle \text{path } \sigma \rangle \langle k \rangle \langle n \rangle$ , OF path-is-path  $\langle k < n \rangle$  ncdk *] by auto
qed

define i where  $\langle i == \text{LEAST } j. j \in \{k..n\} \wedge \text{path } \sigma j = \text{ipd } (\text{path } \sigma k) \rangle$ 
  have iipd:  $\langle i \in \{k..n\} \rangle \langle \text{path } \sigma i = \text{ipd } (\text{path } \sigma k) \rangle$  unfolding i-def using LeastI-ex[OF ipdex] by
simp-all
have **:  $\langle \forall i' \in \{k..<i'\}. \text{path } \sigma i' \neq \text{ipd } (\text{path } \sigma k) \rangle$  proof (rule, rule ccontr)
  fix i' assume *:  $\langle i' \in \{k..<i'\} \rangle \langle \neg \text{path } \sigma i' \neq \text{ipd } (\text{path } \sigma k) \rangle$ 
  hence **:  $\langle i' \in \{k..n\} \wedge \text{path } \sigma i' = \text{ipd } (\text{path } \sigma k) \rangle$  (is  $\langle ?P i' \rangle$ ) using iipd(1) by auto
  thus  $\langle \text{False} \rangle$  using Least-le[of  $\langle ?P \rangle \langle i' \rangle$ ] i'-def * by auto
qed
have  $\langle i \neq k \rangle$  using iipd(2) by (metis nret path-in-nodes ipd-not-self)
hence  $\langle k < i \rangle$  using iipd(1) by simp
hence  $\langle \text{cs}^{\text{path}} \sigma i = [n \leftarrow \text{cs}^{\text{path}} \sigma k . \text{ipd } n \neq \text{path } \sigma i] @ [\text{path } \sigma i] \rangle$  using cs-ipd[OF iipd(2)] **] by fast
hence csi:  $\langle \text{cs}^{\text{path}} \sigma i = \text{cs}^{\text{path}} \sigma' i' \rangle$  using csi' csk unfolding iipd'(2) iipd(2) by (metis last-cs)
hence  $\langle (\text{LEAST } i'. k' < i' \wedge (\exists i. \text{cs}^{\text{path}} \sigma i = \text{cs}^{\text{path}} \sigma' i')) \leq i' \rangle$  (is  $\langle (\text{LEAST } x. ?P x) \leq \rightarrow \rangle$ )
  using  $\langle k' < i' \rangle$  Least-le[of  $\langle ?P \rangle \langle i' \rangle$ ] by blast
hence nw:  $\langle \forall j' \in \{i'..<n'\}. v \notin \text{writes } (\text{path } \sigma' j') \rangle$  using dcd(7) allB-atLeastLessThan-lower by blast
moreover have  $\langle v \in \text{writes } (\text{path } \sigma' j') \rangle$  using nddj' unfolding is-ddi-def by auto
moreover have  $\langle i' \leq j' \rangle$  using iipd'(1) by auto
ultimately have  $\langle \text{False} \rangle$  using  $\langle j' < n' \rangle$  by auto
thus  $\langle ?thesis \rangle$  ..
qed
qed
next
assume  $\langle \neg (\exists j' \in \{k'..<n'\}. v \in \text{writes } (\text{path } \sigma' j')) \rangle$ 
  hence  $\langle n' \text{dcd}^{\text{path}} \sigma', v \rightarrow k' \text{ via } (\text{path } \sigma) k \rangle$  unfolding is-dcdi-via-def using dcd(2-4) csk  $\langle k' < n' \rangle$ 
path-is-path by metis
  thus  $\langle ?thesis \rangle$  using dcd.IH scp.intros(4) by blast
qed
with 1 show  $\langle ?case \rangle$  ..
qed

```

```

theorem cop-in-scop: assumes  $\langle ((\sigma, k), (\sigma', k')) \in \text{cop} \rangle$  shows  $\langle (\text{path } \sigma, k) \in \text{scop} \wedge (\text{path } \sigma', k') \in \text{scp} \rangle$ 
using assms

```

```

apply (induct rule: cop.induct)
  apply (simp add: cp-in-scp)
using cp-in-scp scop.intros scp.intros(2)
  apply blast
using cp-in-scp scop.intros scp.intros(2)
apply blast
done

```

The main correctness result for out single execution approximation follows directly.

```

theorem scop-correct: assumes  $\langle scop = empty \rangle$  shows  $\langle secure \rangle$ 
  using scop-correct assms cp-in-scp by fast

```

**end**

**end**

### 3 Example: Program Dependence Graphs

Program dependence graph (PDG) based slicing provides a very crude but direct approximation of our characterisation. As such we can easily derive a corresponding correctness result.

```

theory PDG imports IFC
begin

```

```

context IFC
begin

```

We utilise our established dependencies on program paths to define the PDG. Note that PDGs usually only contain immediate control dependencies instead of the transitive ones we use here. However as slicing is considering reachability questions this does not affect the result.

```

inductive-set pdg where
 $\langle [i \text{ cd}^\pi \rightarrow k] \implies (\pi k, \pi i) \in pdg \rangle \mid$ 
 $\langle [i \text{ dd}^{\pi,v} \rightarrow k] \implies (\pi k, \pi i) \in pdg \rangle$ 

```

The set of sources is the set of nodes reading high variables.

```

inductive-set sources where
 $\langle n \in nodes \implies h \in hvars \implies h \in reads \ n \implies n \in sources \rangle$ 

```

The forward slice is the set of nodes reachable in the PDG from the set of sources. To ensure security slicing aims to prove that no observable node is contained in the

```

inductive-set slice where
 $\langle n \in sources \implies n \in slice \rangle \mid$ 
 $\langle m \in slice \implies (m,n) \in pdg \implies n \in slice \rangle$ 

```

As the PDG does not contain data control dependencies themselves we have to decompose these.

```

lemma dcd-pdg: assumes  $\langle n \text{ dcd}^{\pi,v} \rightarrow m \text{ via } \pi' m' \rangle$  obtains l where  $\langle (\pi m, l) \in pdg \rangle$  and  $\langle (l, \pi n) \in pdg \rangle$ 
proof –

```

```

  assume r:  $\langle (\bigwedge l. (\pi m, l) \in pdg \implies (l, \pi n) \in pdg \implies thesis) \rangle$ 

```

```

  obtain l' n' where ln:  $\langle cs^\pi m = cs^{\pi'} m' \wedge cs^\pi n = cs^{\pi'} n' \wedge n' \text{ dd}^{\pi',v} \rightarrow l' \wedge l' \text{ cd}^{\pi'} \rightarrow m' \rangle$  using assms

```

```

unfolding is-dcdi-via-def by metis

```

```

  hence mn:  $\langle \pi' m' = \pi m \wedge \pi' n' = \pi n \rangle$  by (metis last-cs ln)

```

```

  have 1:  $\langle (\pi m, \pi' l') \in pdg \rangle$  by (metis ln mn pdg.intros(1))

```

```

  have 2:  $\langle (\pi' l', \pi n) \in pdg \rangle$  by (metis ln mn pdg.intros(2))

```

```

  show thesis using 1 2 r by auto
qed

```

By induction it directly follows that the slice is an approximation of the single critical paths.

```

lemma scp-slice:  $\langle (\pi, i) \in scp \implies \pi i \in slice \rangle$ 
  apply (induction rule: scp.induct)
  apply (simp add: path-in-nodes slice.intros(1) sources.intros)
  using pdg.intros(1) slice.intros(2) apply blast
  using pdg.intros(2) slice.intros(2) apply blast
  by (metis dcd-pdg slice.intros(2))

```

```

lemma scop-slice:  $\langle (\pi, i) \in scop \implies \pi i \in slice \cap dom(att) \rangle$  by (metis IntI scop.cases scp-slice)

```

The requirement targeted by slicing, that no observable node is contained in the slice, is thereby a sound criteria for security.

```

lemma pdg-correct: assumes  $\langle slice \cap dom(att) = \{\} \rangle$  shows  $\langle secure \rangle$ 
proof (rule ccontr)
  assume  $\langle \neg secure \rangle$ 
  then obtain  $\pi i$  where  $\langle (\pi, i) \in scop \rangle$  using scop-correct by force
  thus  $\langle False \rangle$  using scop-slice assms by auto
qed

```

end

end

## References

- [1] A. Bohannon, B. C. Pierce, V. Sjöberg, S. Weirich, and S. Zdancewic. Reactive noninterference. In *Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS '09*, pages 79–90, New York, NY, USA, 2009. ACM.