# 1 VCG Completeness

**theory** *VCG-Completeness* = *VCG*:

— In completeVCG we show that verification conditions are tautologies. If the safety logic is complete, this means they are provable.

**locale** *completeVCG* = *VCG* +

**fixes** *Starters*::'*program* ⇒ ('*pos* × '*mem*) *set*
**defines** *Starters* Π ≡ {*s*. Π,*s* ⊨ *initF* Π ∨ (∃ *A*. *anF* Π (*fst s*) = *Some A* ∧ Π,*s* ⊨ *A* ∧ Π,*s* ⊨ *safeF* Π (*fst s*))}

**fixes** *branch* :: '*program* ⇒ (('*pos* × '*mem*) × ('*pos* × '*mem*)) *set*
**defines** *branch* Π ≡ {(*s*,*s*′). ∃ *B*. (*fst s*′,*B*) ∈ *set* (*succsF* Π (*fst s*)) ∧ Π,*s* ⊨ *B*}

**fixes** *effS*$_B$:: '*program* ⇒ (('*pos* × '*mem*) × ('*pos* × '*mem*)) *set*
**defines** *effS*$_B$ Π ≡ (*effS* Π) ∩ (*branch* Π)

**fixes** *strongAn*:: '*program* ⇒ *bool*
**defines** *strongAn* Π ≡ (∀ *s* ∈ *ReachableFrom* (*effS*$_B$ Π) (*Starters* Π). Π,*s* ⊨ *aF* Π (*fst s*) ∧ Π,*s* ⊨ *safeF* Π (*fst s*))

**assumes** *semImpI*: ⟦ *wf* Π; Π,*s* ⊨ *f* ⟹ Π,*s* ⊨ *f*′ ⟧ ⟹ Π,*s* ⊨ *f* ↛ *f*′

**assumes** *succsFprogress*: ⟦ *wf* Π; Π,(*p*,*m*) ⊨ *B*;
(*p*′,*B*) ∈ *set* (*succsF* Π *p*′′) ⟧
⟹ *p*=*p*′′ ∧ (∃ *m*′. ((*p*,*m*),(*p*′,*m*′)) ∈ (*effS* Π))

**assumes** *completeWpF*: ⟦ *wf* Π;
(*p*′,*B*) ∈ *set* (*succsF* Π *p*); Π,(*p*,*m*) ⊨ *B*;
((*p*,*m*),(*p*′,*m*′)) ∈ (*effS* Π);
Π,(*p*′,*m*′) ⊨ *Q* ⟧ ⟹ Π,(*p*,*m*) ⊨ *wpF* Π *p p*′ *Q*

**assumes** *ipc-domC*:
*wf* Π ⟹ *ipc* Π ∈ *set* (*domC* Π)

**assumes** *succsF-domC*:
⟦ *wf* Π; (*p*′,*B*) ∈ *set* (*succsF* Π *p*) ⟧ ⟹ (*p* ∈ *set* (*domC* Π)) ∧ (*p*′ ∈ *set* (*domC* Π))

**lemma** (**in** *completeVCG*) *ReachableFrom-isafeF-aux*: ⋀ *S*. ⟦ *wf* Π; *strongAn* Π ⟧ ⟹
(∀ *s* ∈ *ReachableFrom* (*effS*$_B$ Π) (*Starters* Π).
((∃ *V*. *noAnn V* (*anF* Π) ∧ *set V* ∪ *set S* = *set* (*domC* Π) ∧ *set V* ∩ *set S* = {} ∧ (*V* ≠ [] ⟶ *V*@[*fst s*] ∈ (*paths* (*succsF* Π))))

$\longrightarrow$ (($fst\ s \in set\ (domC\ \Pi)$) $\longrightarrow$ ($\Pi$,$s \models isafeF\ \Pi\ (fst\ s)$)))))
— S contains positions not yet visited by isafeF. It decreases as isafeF expands.
**theorem** (**in** *completeVCG*) *ReachableFrom-isafeF*:
$[\![\ wf\ \Pi;\ strongAn\ \Pi\ ]\!] \Longrightarrow (\forall\, s \in ReachableFrom\ (effS_B\ \Pi)\ (Starters\ \Pi).\ (fst\ s \in set\ (domC\ \Pi)) \longrightarrow \Pi,s \models isafeF\ \Pi\ (fst\ s))$
**theorem** (**in** *completeVCG*) *vcg-tautology*:
**assumes** *wf-Pi*: *wf* $\Pi$
**assumes** *strongAn*: *strongAn* $\Pi$
**shows** $\Pi,s \models vcg\ \Pi$
**theorem** (**in** *completeVCG*) *vcg-completeness*:
$[\![\ wf\ \Pi;\ strongAn\ \Pi;\ (\forall\ s.\ \Pi,s \models vcg\ \Pi) \longrightarrow \Pi \vdash vcg\ \Pi\ ]\!] \Longrightarrow \Pi \vdash vcg\ \Pi$

**lemma** *ReachableFrom-trans*:
$[\![\ a \in ReachableFrom\ R\ S;\ b \in ReachableFrom\ R\ \{a\}\ ]\!] \Longrightarrow b \in ReachableFrom\ R\ S$
**apply** (*simp add*: *ReachableFrom-conv*)
**apply** *fastsimp*
**done**

**consts** *ReachableFrom-k*:: $('a \times 'a)\ set \Rightarrow 'a\ set \Rightarrow (nat \times 'a)\ set$
**inductive** *ReachableFrom-k R S*
**intros**
*init*: $a \in S \Longrightarrow (0,a) \in ReachableFrom\text{-}k\ R\ S$
*step*: $(k,a) \in ReachableFrom\text{-}k\ R\ S \Longrightarrow (a,b) \in R \Longrightarrow (k+1,b) \in ReachableFrom\text{-}k\ R\ S$

**constdefs** *ReachableFromIn*:: $('a \times 'a)\ set \Rightarrow 'a\ set \Rightarrow nat \Rightarrow 'a\ set$
$ReachableFromIn\ R\ S\ k \equiv \{a.\ (k,a) \in ReachableFrom\text{-}k\ R\ S\}$

**lemma** *k-init*: $a \in S \Longrightarrow a \in ReachableFromIn\ R\ S\ 0$
**apply** (*simp add*: *ReachableFromIn-def*)
**apply** (*rule init*)
**apply** *simp*
**done**

**lemma** *k-step*: $a \in ReachableFromIn\ R\ S\ k \Longrightarrow (a,b) \in R \Longrightarrow b \in ReachableFromIn\ R\ S\ (k+1)$
**apply** (*simp add*: *ReachableFromIn-def*)
**apply** (*drule step*)
**apply** *simp*

**apply** *simp*
**done**

**consts**
*ispecFe*:: (*'program* × *'pos* × *nat* × (*'program* ⇒ *'pos* ⇒ *'form option*) × *'form* × (*'form list* ⇒ *'form*) × (*'form* ⇒ *'form* ⇒ *'form*) ×
( *'program* ⇒ *'pos* ⇒ *'form*) × (*'program* ⇒ *'pos* ⇒ (*'pos* × *'form*) *list*) × (*'program* ⇒ *'pos* ⇒ *'pos* ⇒ *'form* ⇒ *'form*)) ⇒ *'form*

**recdef**
*ispecFe* *measure* (λ *a*. (*fst* (*snd* (*snd a*))))
*ispecFe* (Π,*p*,*k*,*specF*,*TrueF*,*Conj*,*Impl*,*safeF*,*succsF*,*wpF*) =
(*case specF* Π *p*
*of Some Q* ⇒ *Q*
| *None* ⇒ (*case k*
*of 0* ⇒ *TrueF*
| *Suc k′* ⇒ *Conj* (*map* (λ(*p′*,*B*). *Impl B* (*wpF* Π *p p′* (*Conj* [*safeF* Π *p′*, *ispecFe* (Π,*p′*,*k′*,*specF*,*TrueF*,*Conj*,*Impl*,*safeF*,*succsF*,*wpF*)])))) (*succsF* Π *p*))))

**locale** *exprSL* = *completeVCG* +

**fixes** *finals*:: *'program* ⇒ *'pos set*
**defines**
*finals* Π == {*p*. *p* ∈ *set* (*domC* Π) ∧ ¬ (∃ *p′ B*. (*p′*,*B*) ∈ *set* (*succsF* Π *p*))}

**fixes** *specF*:: *'program* ⇒ *'pos* ⇒ *'form option*

**fixes** *ispecF*:: *'program* ⇒ *'pos* ⇒ *nat* ⇒ *'form*
**defines** *ispecF* ≡ λΠ *p k*. *ispecFe* (Π,*p*,*k*,*specF*,*TrueF*,*Conj*,*Impl*,*safeF*,*succsF*,*wpF*)

**assumes** *correctWpF*:
⟦ *wf* Π; (*p′*,*B*) ∈ *set* (*succsF* Π *p*); Π,(*p*,*m*) ⊨ *B*; ((*p*,*m*),(*p′*,*m′*)) ∈ (*effS* Π);
Π,(*p*,*m*) ⊨ *wpF* Π *p p′ Q* ⟧⟹ Π,(*p′*,*m′*) ⊨ *Q*

**assumes** *specF-pos*: ∀ *p*. *p* ∈ {*ipc* Π}∪(*finals* Π) = (*specF* Π *p* ≠ *None*)

**assumes** *specF-ipc*: *specF* Π (*ipc* Π) = *Some* (*initF* Π)

**assumes** *ipc-not-succsF*: *wf* Π ⟹ ∀ *p*. *ipc* Π ∉ *set* (*map fst* (*succsF* Π *p*))

**lemma** (**in** *exprSL*) *ispecFe-def*:
*ispecFe* (Π, *p*, *k*, *specF*, *TrueF*, *Conj*, *Impl*, *safeF*, *succsF*, *wpF*) =
(*case specF* Π *p of*
*None* ⇒ *case k of 0* ⇒ *TrueF*
| *Suc k′* ⇒
*Conj* (*map* (λ(*p′*, *B*). *Impl B* (*wpF* Π *p p′* (*Conj* [*safeF* Π *p′*, *ispecFe* (Π, *p′*, *k′*, *specF*,

*TrueF*, *Conj*, *Impl*, *safeF*, *succsF*, *wpF*)])))
(*succsF* Π *p*))
| *Some Q* ⇒ *Q*)
**apply** (*subst ispecFe.simps*)
**apply** *simp*
**done**

**lemma** (**in** *exprSL*) *ispecF-def*:
⋀ *p*. *ispecF* Π *p k* = (*case specF* Π *p*
*of Some Q* ⇒ *Q*
| *None* ⇒ (*case k of 0* ⇒ ⌞*True*⌟
| *Suc k'* ⇒ ⌞⋀⌟ (*map* (λ(*p'*,*B*). *B* ⌞⇒⌟ (*wpF* Π *p p'* (⌞⋀⌟ [*safeF* Π *p'*, *ispecF* Π *p'*
*k'*])))) (*succsF* Π *p*)))
**apply** (*induct-tac k*)
**apply** (*simp add*: *ispecF-def*)
**apply** (*case-tac specF* Π *p*)
**apply** *simp*

**apply** *simp*

**apply** (*case-tac specF* Π *p*)
**apply** (*simp add*: *ispecF-def*)

**apply** (*simp add*: *ispecF-def*)
**done**

**lemma** *ReachableFromIn-elims*:
⟦ *s* ∈ *ReachableFromIn R S k*; ⟦ *k* = *0*; *s* ∈ *S* ⟧ ⟹ *P*;
⋀*s' k'*. ⟦*k* = *Suc k'*; *s'* ∈ *ReachableFromIn R S k'*; (*s'*, *s*) ∈ *R*⟧ ⟹ *P*⟧ ⟹ *P*
**apply** (*simp add*: *ReachableFromIn-def*)
**apply** (*erule ReachableFrom-k.elims*)
**apply** *simp*

**apply** *simp*
**done**

**lemma** *ReachableFrom-ReachableFromIn*:
*s* ∈ *ReachableFrom R S* ⟹ ∃ *k*. *s* ∈ *ReachableFromIn R S k*
**apply** (*erule ReachableFrom.induct*)
**apply** (*drule-tac R=R* **in** *k-init*)
**apply** *fastsimp*

**apply** (*erule exE*)
**apply** (*rule-tac x=k+1* **in** *exI*)
**apply** (*rule k-step*)

**apply** *simp*
**apply** *simp*
**done**

**lemma** *ReachableFromIn-ReachableFrom*:
$s \in$ *ReachableFromIn R S k* $\Longrightarrow s \in$ *ReachableFrom R S*
**apply** (*simp add*: *ReachableFromIn-def*)
**apply** (*erule ReachableFrom-k.induct*)
**apply** (*rule ReachableFrom.init*)
**apply** *simp*

**apply** (*rule ReachableFrom.step*)
**apply** *simp*
**apply** *simp*
**done**

**lemma** *k-lstep*:
$[\![ s' \in$ *ReachableFromIn R S k* $]\!] \Longrightarrow k > 0 \longrightarrow (\exists\ s\ s''.\ s \in S \wedge (s,s'') \in R \wedge s' \in$ *ReachableFromIn R* $\{s''\}$ $(k - 1))$
**apply** (*simp add*: *ReachableFromIn-def*)
**apply** (*erule ReachableFrom-k.induct*)
**apply** *simp*
**apply** (*rule impI* | *erule conjE*)+
**apply** (*case-tac k*)
**apply** *simp*
**apply** (*erule ReachableFrom-k.elims*)
**apply** *simp*
**apply** (*rule-tac x*=*aa* **in** *exI*)
**apply** *simp*
**apply** (*rule-tac x*=*b* **in** *exI*)
**apply** *simp*
**apply** (*rule ReachableFrom-k.init*)
**apply** *simp*

**apply** *simp*

**apply** *simp*
**apply** (*erule conjE* | *erule exE*)+
**apply** (*rule-tac x*=*s* **in** *exI*)
**apply** *simp*

**apply** (*rule-tac x*=$s''$ **in** *exI*)
**apply** *simp*
**apply** (*drule-tac k*=*nat* **and** *a*=*a* **in** *ReachableFrom-k.step*)

**apply** *simp*

**apply** *simp*
**done**

**theorem** (**in** *exprSL*) *strongAn-exist*:
**assumes** *wf-Pi*: *wf* Π
**and** *bounded-diam*: ∀ *s s′*. *s′* ∈ *ReachableFrom* (*effS*$_B$ Π) {*s*} ⟶ (∃ *r*. *r* ≤ *d* ∧ *s′* ∈ *ReachableFromIn* (*effS*$_B$ Π) {*s*} *r*)
**and** *isSafe*: ∀ *s* ∈ *ReachableFrom* (*effS*$_B$ Π) {*s*. Π,*s* ⊨ *initF* Π}. Π,*s* ⊨ *safeF* Π (*fst s*)
**and** *correct-specF*: ∀ *s* ∈ *ReachableFrom* (*effS*$_B$ Π) {*s*. Π,*s* ⊨ *initF* Π}. (∀ *Q*. *specF* Π (*fst s*) = *Some* *Q* ⟶ Π,*s* ⊨ *Q*)
**and** *anF-ispecF*: ∀ *p*. *anF* Π *p* = *Some* (*ispecF* Π *p d*)
**shows** *strongAn* Π
**proof** −

**have** *ispecF-safeF-ReachableFromIn*: ⋀*k s*. Π,*s* ⊨ *ispecF* Π (*fst s*) *k* ⟹ Π,*s* ⊨ *safeF* Π (*fst s*) ⟹ *fst s* ≠ *ipc* Π ⟹ (∀ *l*. *l*≤*k* ⟶ (∀ *s′* ∈ *ReachableFromIn* (*effS*$_B$ Π) {*s*} *l*. Π,*s′* ⊨ *ispecF* Π (*fst s′*) (*k*−*l*) ∧ Π,*s′* ⊨ *safeF* Π (*fst s′*))) (**is** ⋀ *z*. *PROP ?P z*)
  **proof** −
  **fix** *z*
  **show** *PROP ?P z*
  **proof** (*induct z rule*: *nat-less-induct*, *intro allI impI ballI*)
    **fix** *k s l s′*
    **assume** *hyp*: ∀ *k′*<*k*. ∀ *s*. Π,*s* ⊨ *ispecF* Π (*fst s*) *k′* ⟶ Π,*s* ⊨ *safeF* Π (*fst s*) ⟶ *fst s* ≠ *ipc* Π ⟶ (∀ *l*≤*k′*. (∀ *s′*∈*ReachableFromIn* (*effS*$_B$ Π) {*s*} *l*. Π,*s′* ⊨ *ispecF* Π (*fst s′*) (*k′* − *l*) ∧ Π,*s′* ⊨ *safeF* Π (*fst s′*)))
    **assume** *s-ispecF-k*: Π,*s* ⊨ *ispecF* Π (*fst s*) *k*
    **assume** *s-safeF*: Π,*s* ⊨ *safeF* Π (*fst s*)
    **assume** *s-not-ipc*: *fst s* ≠ *ipc* Π
    **assume** *l-k*: *l* ≤ *k*
    **assume** *s′-ReachableFromIn-l*: *s′* ∈ *ReachableFromIn* (*effS*$_B$ Π) {*s*} *l*
    **show** Π,*s′* ⊨ *ispecF* Π (*fst s′*) (*k* − *l*) ∧ Π,*s′* ⊨ *safeF* Π (*fst s′*) (**is** *?s′-ispecF*)
      **proof** (*cases l*)
        **case** *0*
        **from** *0 s′-ReachableFromIn-l*
        **have** *s-eq-s′*: *s* = *s′*
          **apply** −
          **apply** (*erule ReachableFromIn-elims*)
          **apply** *simp*
          **apply** *simp*
          **done**
        **from** *0 s-eq-s′ s-ispecF-k s-safeF* **show** *?s′-ispecF*
          **by** *simp*

**next**
**case** (*Suc l'*)

**from** *Suc* **have** *l'-l*: $l = Suc\ l'$
**by** *simp*

**from** *l'-l s'-ReachableFromIn-l*
**obtain** *s''* **where** *s'-ReachableFromIn-l'*: $s' \in ReachableFromIn\ (effS_B\ \Pi)\ \{s''\}\ l'$
**and** *s-s''-effSB*: $(s,s'') \in (effS_B\ \Pi)$
**apply** −
**apply** (*drule k-lstep*)
**apply** *fastsimp*
**done**

**from** *l'-l l-k* **have** *l'-k*: $l' < k$
**by** *arith*

**from** *l'-k*
**obtain** *k'* **where** *k'-k*: $k = Suc\ k'$
**apply** −
**apply** (*case-tac k*::*nat*)
**by** *arith*

**from** *k'-k l'-k* **have** *l'-k'*: $l' \leq k'$
**by** *arith*

**from** *k'-k l'-l* **have** *k-l-k'-l'*: $k - l = (k' - l')$
**by** *arith*

**obtain** *p m* **where** *s-def*: $s = (p,m)$
**by** *fastsimp*

**obtain** *p' m'* **where** *s'-def*: $s' = (p',m')$
**by** *fastsimp*

**obtain** *p'' m''* **where** *s''-def*: $s'' = (p'',m'')$
**by** *fastsimp*

**from** *s-s''-effSB s-def s''-def*
**obtain** *B* **where** *p''-B-succsF-p*: $(p'',B) \in set\ (succsF\ \Pi\ p)$
**and** *s-B*: $\Pi,s \models B$ **and** *s-s''-effS*: $(s,s'') \in effS\ \Pi$
**apply** −
**apply** (*simp add*: *effS*$_B$*-def branch-def*)
**apply** *fastsimp*
**done**

**from** *p''-B-succsF-p*
**have** *p-not-final*: $p \notin \mathit{finals}\ \Pi$
**by** (*fastsimp simp add*: *finals-def*)

**from** *s-not-ipc s-def p-not-final specF-pos*
**have** *p-specF*: $\mathit{specF}\ \Pi\ p = \mathit{None}$
**by** *fastsimp*

**from** *s-ispecF-k k'-k* **have** *s''-safeF-ispecF*: $\Pi,s'' \models \bigwedge [\mathit{safeF}\ \Pi\ p'', \mathit{ispecF}\ \Pi\ p''\ k']$
**proof** −

**from** *s-ispecF-k p-specF s-def k'-k p''-B-succsF-p s''-def*
**have** *s-B-Imp-wpF*: $\Pi,s \models B \Rightarrow \mathit{wpF}\ \Pi\ p\ p''\ (\bigwedge [\mathit{safeF}\ \Pi\ p'', \mathit{ispecF}\ \Pi\ p''\ k'])$
**apply** −
**apply** (*drule-tac* $P{=}\lambda\ \mathit{ispecF}.\ \Pi,s \models \mathit{ispecF}$ **in** *subst*[*OF ispecF-def*])
**apply** (*simp add*: *semConj*)
**apply** (*erule-tac* $x{=}(p'',B)$ **in** *ballE*)
**apply** (*simp add*: *split-def*)
**apply** *simp*
**done**

**from** *s-B s-B-Imp-wpF*
**have** *s-wpF*: $\Pi,s \models \mathit{wpF}\ \Pi\ p\ p''\ (\bigwedge [\mathit{safeF}\ \Pi\ p'', \mathit{ispecF}\ \Pi\ p''\ k'])$
**apply** −
**apply** (*drule semImpE*)
**apply** *simp*
**done**

**from** *wf-Pi s-wpF s-B p''-B-succsF-p s-def s''-def s-s''-effS s-wpF*
**show** *?thesis*
**apply** −
**apply** (*simp only*:)
**apply** (*rule correctWpF*)
**apply** *simp+*
**done**
**qed**

**from** *s''-safeF-ispecF s''-def*
**have** *s''-ispecF*: $\Pi,s'' \models \mathit{ispecF}\ \Pi\ (\mathit{fst}\ s'')\ k'$
**by** (*simp add*: *semConj*)

**from** *s''-safeF-ispecF s''-def*
**have** *s''-safeF*: $\Pi,s'' \models \mathit{safeF}\ \Pi\ (\mathit{fst}\ s'')$
**by** (*simp add*: *semConj*)

**from** *p''-B-succsF-p wf-Pi ipc-not-succsF*[**where** Π=Π] **have** *p''-not-ipc*: $p'' \neq ipc\ \Pi$
**apply** −
**apply** *simp*
**apply** (*erule-tac x=p* **in** *allE*)
**apply** *clarsimp*
**apply** (*simp add*: *in-set-conv-decomp*)
**apply** (*erule exE* | *erule conjE*)+
**apply** *simp*
**done**

**from** *k'-k* **have** *k'-less-k*: $k' < k$
**by** *arith*

**from** *hyp s''-ispecF s''-safeF k'-less-k p''-not-ipc l'-k' s'-ReachableFromIn-l' s''-def*
**have** *s'-ispecF-k'-l'-safeF*: $\Pi,s' \models ispecF\ \Pi\ (fst\ s')\ (k' - l') \wedge \Pi,s' \models safeF\ \Pi\ (fst\ s')$
**by** *fastsimp*

**from** *s'-ispecF-k'-l'-safeF k-l-k'-l'*
**show** *?s'-ispecF*
**by** *simp*
**qed**
**qed**
**qed**

**have** *ReachableFromIn-ispecF*: $\bigwedge k\ s.\ (\forall l.\ l \leq k \longrightarrow (\forall s' \in ReachableFromIn\ (effS_B\ \Pi)\ \{s\}\ l.\ \Pi,s' \models ispecF\ \Pi\ (fst\ s')\ (k-l))) \Longrightarrow fst\ s \neq ipc\ \Pi \Longrightarrow \Pi,s \models ispecF\ \Pi\ (fst\ s)\ k$
**proof** −
**fix** *k s*
**assume** *l-k-ReachableFromIn-ispecF*: $\forall l \leq k.\ \forall s' \in ReachableFromIn\ (effS_B\ \Pi)\ \{s\}\ l.\ \Pi,s' \models ispecF\ \Pi\ (fst\ s')\ (k - l)$
**assume** *s-not-ipc*: $fst\ s \neq ipc\ \Pi$
**have** *s-ReachableFromIn-0*: $s \in ReachableFromIn\ (effS_B\ \Pi)\ \{s\}\ 0$
**apply** −
**apply** (*rule k-init*)
**by** *simp*

**from** *l-k-ReachableFromIn-ispecF s-ReachableFromIn-0*
**show** $\Pi,s \models ispecF\ \Pi\ (fst\ s)\ k$
**by** *fastsimp*
**qed**

**have** *ispecF-d-inc*: $\bigwedge\ s.\ \Pi,s \models safeF\ \Pi\ (fst\ s) \Longrightarrow \Pi,s \models ispecF\ \Pi\ (fst\ s)\ d \Longrightarrow \Pi,s \models ispecF\ \Pi\ (fst\ s)\ (d+1)$
**proof** −

**fix** $s$
**assume** *s-safeF*: $\Pi,s \models$ *safeF* $\Pi$ (*fst* $s$)
**assume** *s-ispecF-d*: $\Pi,s \models$ *ispecF* $\Pi$ (*fst* $s$) $d$
**show** $\Pi,s \models$ *ispecF* $\Pi$ (*fst* $s$) ($d$+*1*)
**proof** (*cases specF* $\Pi$ (*fst* $s$))
  **case** (*Some* $Q$)
  **from** *Some s-ispecF-d* **show** *?thesis*
    **by** (*simp add*: *ispecF-def*)
**next**
  **case** *None*

  **obtain** $p$ $m$ **where** *s-def*: $s = (p,m)$
    **by** *fastsimp*

  **from** *None s-def specF-pos*[**where** $\Pi$=$\Pi$]
  **have** *p-not-ipc*: $p \neq ipc\ \Pi$
    **apply** −
    **apply** (*erule-tac* $x$=$p$ **in** *allE*)
    **apply** *simp*
    **done**

  **from** *None s-def specF-pos*[**where** $\Pi$=$\Pi$]
  **have** *p-not-final*: $p \notin finals\ \Pi$
    **apply** −
    **apply** (*erule-tac* $x$=$p$ **in** *allE*)
    **apply** *simp*
    **done**

  **from** *s-ispecF-d s-safeF s-def p-not-ipc*
  **have** *ReachableFromIn-ispecF-d*: $\forall\, l \leq d.\ \forall\, s' \in ReachableFromIn\ (effS_B\ \Pi)\ \{s\}\ l.\ \Pi,s' \models ispecF\ \Pi\ (fst\ s')\ (d - l) \wedge \Pi,s' \models safeF\ \Pi\ (fst\ s')$
    **apply** −
    **apply** (*drule ispecF-safeF-ReachableFromIn*)
    **apply** *simp*
    **apply** *simp*
    **apply** *simp*
    **done**

  **have** *ReachableFromIn-ispecF-inc-d*:
  $\bigwedge k\ l\ s'.\ k = (d + 1) - l \longrightarrow l \leq d + 1 \longrightarrow s' \in ReachableFromIn\ (effS_B\ \Pi)\ \{s\}\ l \longrightarrow \Pi,s' \models ispecF\ \Pi\ (fst\ s')\ ((d + 1) - l) \wedge \Pi,s' \models safeF\ \Pi\ (fst\ s')$ (**is** $\bigwedge k.\ PROP\ ?P\ k$)
    **proof** −
    **fix** $z$
    **show** *PROP ?P z*

**proof** (*induct z rule*: *nat-less-induct*, *intro impI*)
**fix** $k\ l\ s'$
**assume** *Hyp*: $\forall k'<k.\ \forall l'\ s'.\ k' = d + 1 - l' \longrightarrow$
$l' \leq d + 1 \longrightarrow s' \in ReachableFromIn\ (effS_B\ \Pi)\ \{s\}\ l' \longrightarrow \Pi,s' \models ispecF\ \Pi\ (fst\ s')\ (d + 1 - l') \wedge \Pi,s' \models safeF\ \Pi\ (fst\ s')$
**assume** *k-d-l*: $k = (d + 1) - l$
**assume** *l-d*: $l \leq (d + 1)$
**assume** *s'-ReachableFromIn-l*: $s' \in ReachableFromIn\ (effS_B\ \Pi)\ \{s\}\ l$
**show** $\Pi,s' \models ispecF\ \Pi\ (fst\ s')\ (d + 1 - l) \wedge \Pi,s' \models safeF\ \Pi\ (fst\ s')$ (**is** *?s'-ispecF*)
**proof** (*cases k*)
**case** *0*
**from** *0 k-d-l* **have** *l-gr-d*: $l > d$
**by** *arith*

**from** *s'-ReachableFromIn-l*
**have** *s'-ReachableFrom-s*: $s' \in ReachableFrom\ (effS_B\ \Pi)\ \{s\}$
**by** (*rule ReachableFromIn-ReachableFrom*)

**from** *bounded-diam s'-ReachableFrom-s*
**obtain** *r* **where** *r-le-d*: $r \leq d$
**and** *s'-ReachableFromIn-r*: $s' \in ReachableFromIn\ (effS_B\ \Pi)\ \{s\}\ r$
**by** *fastsimp*

**from** *ReachableFromIn-ispecF-d r-le-d s'-ReachableFromIn-r*
**have** *s'-ispecF-d-r-safeF*: $\Pi,s' \models ispecF\ \Pi\ (fst\ s')\ (d - r) \wedge \Pi,s' \models safeF\ \Pi\ (fst\ s')$
**by** *fastsimp*

**show** *?s'-ispecF*
**proof** (*cases specF* $\Pi$ (*fst s'*))
**case** (*Some Q*)
**from** *Some s'-ispecF-d-r-safeF*
**show** *?s'-ispecF*
**by** (*simp add*: *ispecF-def*)
**next**
**case** *None*
**from** *None 0 k-d-l*[*THEN sym*] *s'-ispecF-d-r-safeF* **show** *?s'-ispecF*
**by** (*simp add*: *ispecF-def semTrue*)
**qed**
**next**
**case** (*Suc k'*)
**from** *Suc* **have** *k'-k*: $k = Suc\ k'$
**by** *simp*

**show** *?s'-ispecF*
**proof** (*cases l*)

**case** *0*
**from** *0 s′-ReachableFromIn-l*
**have** *s′-eq-s*: $s' = s$
**apply** −
**apply** (*erule ReachableFromIn-elims*)
**apply** *simp*
**apply** *simp*
**done**

**obtain** $p'$ $m'$ **where** *s′-def*: $s' = (p',m')$
**by** *fastsimp*

**from** *wf-Pi*
**have** *s′-B-wpF*: $\forall\ (p'',B'') \in set\ (succsF\ \Pi\ p').\ \Pi,s' \models B'' \Rightarrow wpF\ \Pi\ p'\ p''\ (\bigwedge [safeF\ \Pi\ p'',\ ispecF\ \Pi\ p''\ k'])$
**proof** (*intro ballI*, *simp add*: *split-paired-all s′-def*, *intro semImpI*, *assumption*)
**fix** $p''$ $B''$
**assume** *p″-B″-succsF-p′*: $(p'',B'') \in set\ (succsF\ \Pi\ p')$
**assume** *s′-B″*: $\Pi,(p',m') \models B''$

**from** *wf-Pi p″-B″-succsF-p′ s′-B″ s′-def*
**obtain** $m''$ **where** *s′-s″-effS*: $((p',m'),(p'',m'')) \in effS\ \Pi$
**apply** −
**apply** (*drule succsFprogress*)
**apply** *simp+*
**apply** (*erule exE*)
**apply** *fastsimp*
**done**

**obtain** $s''$ **where** *s″-def*: $s'' = (p'',m'')$
**by** *fastsimp*

**have** *s″-ispecF-safeF*: $\Pi,(p'',m'') \models ispecF\ \Pi\ p''\ k' \wedge \Pi,(p'',m'') \models safeF\ \Pi\ p''$
**proof** −

**from** *s′-s″-effS s′-def s″-def s′-B″ p″-B″-succsF-p′*
**have** *s′-s″-effSB*: $(s',s'') \in effS_B\ \Pi$
**by** (*fastsimp simp add*: $effS_B$*-def branch-def*)

**from** *0 s′-eq-s s′-ReachableFromIn-l s′-s″-effSB*
**have** *s″-ReachableFromIn-1*: $s'' \in ReachableFromIn\ (effS_B\ \Pi)\ \{s\}\ 1$
**apply** −
**apply** (*drule k-step*)
**apply** *simp*
**apply** *simp*

```
      done

    from k'-k have k'-less-k: k' < k
      by arith

    from Hyp 0 k-d-l[THEN sym] k'-less-k k'-k s''-ReachableFromIn-1 s''-def
    show s''-ispecF-safeF: Π,(p'',m'') ⊨ ispecF Π p'' k' ∧ Π,(p'',m'') ⊨ safeF Π
p''
      apply −
      apply (erule-tac x=k' in allE)
      apply (drule mp, assumption)
      apply (erule-tac x=1 in allE)
      apply (erule-tac x=s'' in allE)
      by fastsimp
  qed

  from s''-ispecF-safeF s''-def
  have s''-safeF-ispecF: Π,(p'',m'') ⊨ ⨅[safeF Π p'', ispecF Π p'' k']
    by (simp add: semConj)

 from wf-Pi p''-B''-succsF-p' s'-B'' s-def s'-def s''-safeF-ispecF s'-s''-effS
 show Π,(p',m') ⊨ wpF Π p' p'' ⨅[safeF Π p'', ispecF Π p'' k']
   apply −
   apply (rule completeWpF)
   apply simp+
   done
qed

from 0 s'-ReachableFromIn-l ReachableFromIn-ispecF-d
have s'-safeF: Π,s' ⊨ safeF Π (fst s')
  by fastsimp

from None s'-eq-s k-d-l[THEN sym] k'-k s'-B-wpF s-def s'-def s'-safeF
show ?s'-ispecF
  apply −
  apply (subst ispecF-def)
  apply (simp add: semConj split-def)
  done

next

  case (Suc l')

  from Suc have l'-l: l = Suc l'
    by simp
```

**from** $k'$-$k$ $k$-$d$-$l$ **have** $l$-$le$-$d$: $l \leq d$
**by** *arith*

**from** *ReachableFromIn-ispecF-d* $l$-$le$-$d$ $s'$-*ReachableFromIn-l*
**have** $s'$-*ispecF-d-safeF*: $\Pi,s' \models$ *ispecF* $\Pi$ (*fst* $s'$) $(d - l) \wedge \Pi,s' \models$ *safeF* $\Pi$ (*fst* $s'$)
**by** *fastsimp*

**show** *?s'-ispecF*
**proof** (*cases specF* $\Pi$ (*fst* $s'$))
**case** (*Some* $Q$)
**from** *Some* $s'$-*ispecF-d-safeF* **show** *?s'-ispecF*
**by** (*simp add*: *ispecF-def*)
**next**
**case** *None*

**obtain** $p'$ $m'$ **where** $s'$-*def*: $s' = (p',m')$
**by** *fastsimp*

**from** *wf-Pi*
**have** $s'$-$B$-$wpF$: $\forall\ (p'',B'') \in set\ (succsF\ \Pi\ p').\ \Pi,s' \models B'' \hookrightarrow wpF\ \Pi\ p'\ p''\ (\bigwedge [safeF\ \Pi\ p'', ispecF\ \Pi\ p''\ k'])$
**proof** (*intro ballI*, *simp add*: *split-paired-all* $s'$-*def*, *intro semImpI*, *assumption*)
**fix** $p''$ $B''$
**assume** $p''$-$B''$-*succsF-p'*: $(p'',B'') \in set\ (succsF\ \Pi\ p')$
**assume** $s'$-$B''$: $\Pi,(p',m') \models B''$

**from** *wf-Pi* $p''$-$B''$-*succsF-p'* $s'$-$B''$ $s'$-*def*
**obtain** $m''$ **where** $s'$-$s''$-*effS*: $((p',m'),(p'',m'')) \in effS\ \Pi$
**apply** $-$
**apply** (*drule succsFprogress*)
**apply** *simp*+
**apply** (*erule exE*)
**apply** *fastsimp*
**done**

**obtain** $s''$ **where** $s''$-*def*: $s'' = (p'',m'')$
**by** *fastsimp*

**have** $s''$-*ispecF-safeF*: $\Pi,(p'',m'') \models ispecF\ \Pi\ p''\ k' \wedge \Pi,(p'',m'') \models safeF\ \Pi\ p''$
**proof** $-$

**from** $s'$-$s''$-*effS* $s'$-*def* $s''$-*def* $s'$-$B''$ $p''$-$B''$-*succsF-p'*
**have** $s'$-$s''$-*effSB*: $(s',s'') \in effS_B\ \Pi$
**by** (*fastsimp simp add*: $effS_B$-*def branch-def*)

```
      have s'-ReachableFromIn-0: s' ∈ ReachableFromIn (effS_B Π) {s'} 0
        apply −
        apply (rule k-init)
        by simp

      from s'-ReachableFromIn-l s'-s''-effSB
      have s''-ReachableFromIn-inc-l: s'' ∈ ReachableFromIn (effS_B Π) {s} (l+1)
        apply −
        apply (drule k-step)
        apply simp
        apply simp
        done

      from k'-k have k'-less-k: k' < k
        by arith

      from k'-k k-d-l have k'-d-l: k' = d − l
        by arith

     from Hyp k-d-l[THEN sym] k'-less-k k'-k k'-d-l l-le-d s''-ReachableFromIn-inc-l
s''-def
      show s''-ispecF-safeF: Π,(p'',m'') ⊨ ispecF Π p'' k' ∧ Π,(p'',m'') ⊨ safeF Π
p''
        apply −
        apply (erule-tac x=k' in allE)
        apply (drule mp, assumption)
        apply (erule-tac x=l + 1 in allE)
        apply (erule-tac x=s'' in allE)
        apply simp
        done
    qed

    from s''-ispecF-safeF s''-def
    have s''-safeF-ispecF: Π,(p'',m'') ⊨ ⨅[safeF Π p'', ispecF Π p'' k']
      by (simp add: semConj)

  from wf-Pi p''-B''-succsF-p' s'-B'' s-def s'-def s''-safeF-ispecF s'-s''-effS
  show Π,(p',m') ⊨ wpF Π p' p'' ⨅[safeF Π p'', ispecF Π p'' k']
    apply −
    apply (rule completeWpF)
    apply simp+
    done
qed

from None s'-def s'-B-wpF k-d-l[THEN sym] k'-k s'-ispecF-d-safeF
```

```
            show ?s'-ispecF
              apply −
              apply (subst ispecF-def)
              apply (simp add: semConj split-def)
              done
          qed


        qed
      qed
    qed
  qed

  have s-ReachableFromIn-0: s ∈ ReachableFromIn (effS_B Π) {s} 0
    apply −
    apply (rule k-init)
    by simp

  from s-ReachableFromIn-0 ReachableFromIn-ispecF-inc-d[where k=d + 1 and l=0]
  show ?thesis
    by simp
 qed
qed


have correct-Starters: ⋀ s. s ∈ Starters Π ⟹ Π,s ⊨ aF Π (fst s) ∧ Π,s ⊨ safeF Π (fst s)
 proof −
  fix s
  assume s-Starters: s ∈ Starters Π
  show Π,s ⊨ aF Π (fst s) ∧ Π,s ⊨ safeF Π (fst s)
   proof (cases Π,s ⊨ initF Π)
    case True
    from True have s-initF: Π,s ⊨ initF Π
      by assumption

    from s-initF have s-ReachableFrom: s ∈ ReachableFrom (effS_B Π) {s. Π,s ⊨ initF Π}
      apply −
      apply (rule ReachableFrom.init)
      by simp


    from s-ReachableFrom isSafe
    have s-safeF: Π,s ⊨ safeF Π (fst s)
      by fastsimp
```

```
      from wf-Pi s-initF have s-ipc: fst s = (ipc Π)
        by (rule initF-ipc)

      from s-ipc specF-pos
      obtain Q where Q-def: specF Π (fst s) = Some Q
        by fastsimp

      from Q-def anF-ispecF
      have aF-Q: aF Π (fst s) = Q
        apply −
        apply (simp add: aF-def ispecF-def)
        done


      from s-ReachableFrom correct-specF Q-def aF-Q
      have s-aF: Π,s ⊨ aF Π (fst s)
        by fastsimp

      from s-safeF s-aF
      show Π,s ⊨ aF Π (fst s) ∧ Π,s ⊨ safeF Π (fst s)
        by simp
    next
      case False
      from False have s-not-initF: ¬ Π,s ⊨ initF Π
        by assumption
      from s-not-initF s-Starters
      obtain A where A-def: anF Π (fst s) = Some A and s-A: Π,s ⊨ A and s-safeF: Π,s ⊨ safeF
Π (fst s)
        apply −
        apply (simp add: Starters-def)
        apply fastsimp
        done

      from A-def s-A s-safeF
      show Π,s ⊨ aF Π (fst s) ∧ Π,s ⊨ safeF Π (fst s)
        by (simp add: aF-def)
    qed
  qed

from anF-ispecF
have in-Starters: ⋀s. Π,s ⊨ aF Π (fst s) ⟹ Π,s ⊨ safeF Π (fst s) ⟹ s ∈ Starters Π
  by (simp add: aF-def Starters-def)
```

**have** *G1*: $\forall\, s \in$ *ReachableFrom* (*effS*$_B$ $\Pi$) (*Starters* $\Pi$). $\Pi$,$s \models$ *aF* $\Pi$ (*fst s*) $\wedge$ $\Pi$,$s \models$ *safeF* $\Pi$ (*fst s*) (**is** *?G1*)
**proof** −
**have** *G2*: $\bigwedge r.\ \forall\, s \in$ *ReachableFromIn* (*effS*$_B$ $\Pi$) (*Starters* $\Pi$) $r$. $\Pi$,$s \models$ *aF* $\Pi$ (*fst s*) $\wedge$ $\Pi$,$s \models$ *safeF* $\Pi$ (*fst s*) (**is** $\bigwedge r.$ *PROP ?P r*)
**proof** −
**fix** $r$
**show** *PROP ?P r*
**proof** (*induct r rule*: *nat-less-induct*)
**case** (*1 n*)
**assume** *Hyp*: $\forall\, m{<}n.\ \forall\, s{\in}$*ReachableFromIn* (*effS*$_B$ $\Pi$) (*Starters* $\Pi$) $m$. $\Pi$,$s \models$ *aF* $\Pi$ (*fst s*) $\wedge$ $\Pi$,$s \models$ *safeF* $\Pi$ (*fst s*)
**show** $\forall\, s{\in}$*ReachableFromIn* (*effS*$_B$ $\Pi$) (*Starters* $\Pi$) $n$. $\Pi$,$s \models$ *aF* $\Pi$ (*fst s*) $\wedge$ $\Pi$,$s \models$ *safeF* $\Pi$ (*fst s*)
**proof** (*intro ballI impI*)
**fix** $s$
**assume** *s-ReachableFromIn-n*: $s \in$ *ReachableFromIn* (*effS*$_B$ $\Pi$) (*Starters* $\Pi$) $n$
**from** *s-ReachableFromIn-n*
**have** *s-ReachableFrom*: $s \in$ *ReachableFrom* (*effS*$_B$ $\Pi$) (*Starters* $\Pi$)
**by** (*rule ReachableFromIn-ReachableFrom*)

**show** $\Pi$,$s \models$ *aF* $\Pi$ (*fst s*) $\wedge$ $\Pi$,$s \models$ *safeF* $\Pi$ (*fst s*) (**is** *?G2'*)
**proof** (*rule ReachableFromIn-elims*[*OF s-ReachableFromIn-n*])
**assume** *n-0*: $n = 0$
**assume** *s-Starters*: $s \in$ *Starters* $\Pi$
**show** *?G2'*
**by** (*rule correct-Starters*)
**next**
**fix** $s'$ $k'$
**assume** *n-k'*: $n =$ *Suc* $k'$
**assume** *s'-ReachableFromIn-k'*: $s' \in$ *ReachableFromIn* (*effS*$_B$ $\Pi$) (*Starters* $\Pi$) $k'$
**assume** *s'-s-effSB*: ($s'$, $s$) $\in$ *effS*$_B$ $\Pi$
**show** *?G2'*
**proof** −

**from** *Hyp n-k' s'-ReachableFromIn-k'*
**have** *s'-aF-safeF*: $\Pi$,$s' \models$ *aF* $\Pi$ (*fst s'*) $\wedge$ $\Pi$,$s' \models$ *safeF* $\Pi$ (*fst s'*)
**by** *fastsimp*

**from** *s'-aF-safeF* **have** *s'-Starters*: $s' \in$ *Starters* $\Pi$
**apply** −
**apply** (*rule in-Starters*)
**apply** *simp*
**apply** *simp*
**done**

**from** $s'$-*s-effSB*
**obtain** $B$ **where** $s'$-*s-succsF*: (*fst* $s$,$B$) ∈ *set* (*succsF* Π (*fst* $s'$)) **and** $s'$-$B$: Π,$s'$ ⊨ $B$ **and** $s'$-*s-effS*: ($s'$,$s$) ∈ *effS* Π
**apply** −
**apply** (*simp add*: *effS*$_B$-*def branch-def*)
**apply** *fastsimp*
**done**

**from** $s'$-*s-succsF*
**have** $s'$-*not-final*: *fst* $s'$ ∉ *finals* Π
**apply** −
**apply** (*simp add*: *finals-def*)
**apply** *fastsimp*
**done**

**from** $s'$-*aF-safeF anF-ispecF*
**have** $s'$-*ispecF*: Π,$s'$ ⊨ *ispecF* Π (*fst* $s'$) (*d*+*1*)
**apply** −
**apply** (*rule ispecF-d-inc*)
**apply** *simp*
**apply** (*simp add*: *aF-def*)
**done**

**show** *?G2'*
**proof** (*cases fst* $s'$ = *ipc* Π)
**case** *False*
**from** *False* $s'$-*not-final specF-pos*
**have** $s'$-*specF-None*: *specF* Π (*fst* $s'$) = *None*
**by** *fastsimp*

**from** $s'$-*ispecF* $s'$-*specF-None* $s'$-*s-succsF*
**have** $s'$-*wp-ispecF*: Π,$s'$ ⊨ $B$ ⇛ *wpF* Π (*fst* $s'$) (*fst* $s$) (⋀ [*safeF* Π (*fst* $s$),*ispecF* Π (*fst* $s$) $d$])
**apply** −
**apply** (*drule-tac* $P$=λ *ispecF*. Π,$s'$ ⊨ *ispecF* **in** *subst*[*OF ispecF-def*])
**apply** (*simp add*: *semConj*)
**apply** (*erule-tac* $x$=(*fst* $s$,$B$) **in** *ballE*)
**apply** *simp*

**apply** *simp*
**done**

**obtain** $p$ $m$ **where** *s-def*: $s$ = ($p$,$m$)
**by** *fastsimp*

**obtain** $p'$ $m'$ **where** $s'$-*def*: $s'$ = ($p'$,$m'$)

**by** *fastsimp*

**from** *wf-Pi s′-s-effS s′-B s′-s-succsF s′-wp-ispecF s-def s′-def*
**have** *s-safeF-ispecF*: $\Pi$,$s \models \bigwedge\!\!\!\bigwedge [safeF\ \Pi\ (fst\ s),\ ispecF\ \Pi\ (fst\ s)\ d]$
**apply** −
**apply** (*simp only*: *fst-conv snd-conv*)
**apply** (*drule semImpE*)
**apply** (*rule correctWpF*)
**apply** *simp*+
**done**

**from** *s-safeF-ispecF anF-ispecF* **show** *?G2′*
**apply** −
**apply** (*simp add*: *aF-def semConj*)
**done**

**next**

**case** *True*
**from** *True* **have** *s′-ipc*: *fst s′* = *ipc* $\Pi$
**by** *assumption*

**from** *s′-ipc specF-ipc*
**have** *specF-s′*: *specF* $\Pi$ (*fst s′*) = *Some* (*initF* $\Pi$)
**by** *simp*

**from** *specF-s′ s′-ispecF*
**have** *s′-initF*: $\Pi$,$s' \models$ (*initF* $\Pi$)
**by** (*simp add*: *ispecF-def*)

**from** *s′-initF*
**have** *s′-ReachableFrom*: $s' \in$ *ReachableFrom* (*effS*$_B$ $\Pi$) {$s$. $\Pi$,$s \models$ *initF* $\Pi$}
**apply** −
**apply** (*rule ReachableFrom.init*)
**apply** *simp*
**done**

**from** *s′-initF s′-s-effSB s′-ReachableFrom*
**have** *s-ReachableFrom*: $s \in$ *ReachableFrom* (*effS*$_B$ $\Pi$) {$s$. $\Pi$,$s \models$ *initF* $\Pi$}
**apply** −
**apply** (*rule ReachableFrom.step*)
**apply** *simp*
**apply** *simp*
**done**

```
                    from s-ReachableFrom isSafe
                    have s-safeF: Π,s ⊨ safeF Π (fst s)
                      by fastsimp

                   have ispecF-trans: ⋀ k. ∀ l. k = d −l ⟶ (∀ s''∈ ReachableFromIn (effS_B Π)
{s} l. Π,s'' ⊨ ispecF Π (fst s'') (d − l)) (is ⋀ k. PROP ?P k)
                    proof −
                    fix k
                    show PROP ?P k
                      proof (induct k rule: nat-less-induct, intro allI impI ballI)
                        fix n l s''
                         assume hyp: ∀ m<n. ∀ l. m = d − l ⟶ (∀ s''∈ReachableFromIn (effS_B
Π) {s} l. Π,s'' ⊨ ispecF Π (fst s'') (d − l))
                        assume n-d-l: n = d − l
                        assume s''-ReachableFromIn-l: s'' ∈ ReachableFromIn (effS_B Π) {s} l
                        show Π,s'' ⊨ ispecF Π (fst s'') (d − l) (is ?G2'')
                          proof (cases specF Π (fst s''))
                            case (Some Q)
                            from Some have ispecF-Q: ispecF Π (fst s'') (d − l) = Q
                              by (simp add: ispecF-def)



                            from s''-ReachableFromIn-l
                            have s''-ReachableFrom-s: s'' ∈ ReachableFrom (effS_B Π) {s}
                              by (rule ReachableFromIn-ReachableFrom)

                            from s-ReachableFrom s''-ReachableFrom-s
                           have s''-ReachableFrom: s'' ∈ ReachableFrom (effS_B Π) {s. Π,s ⊨ initF
Π}
                              apply (rule ReachableFrom-trans)
                              done

                            from correct-specF s''-ReachableFrom Some ispecF-Q
                            show ?G2''
                              by fastsimp
                           next
                            case None
                            from None have specF-s'': specF Π (fst s'') = None
                              by assumption

                            show ?G2''
                              proof (cases d − l)
                                case 0
                                from specF-s'' 0 show ?G2''
```

```
                    apply −
                    apply (simp add:ispecF-def)
                    apply (rule semTrue)
                    done

                  next

                    case (Suc n')
                    have s''-wpF-ispecF: Π,s'' ⊨ ⌊⋀⌋ (map (λ(p',B). B ⌊⇛⌋
                      (wpF Π (fst s'') p' (⌊⋀⌋ [safeF Π p',ispecF Π p' ((d − l) − 1)])))
(succsF Π (fst s'')))
                    proof −
                      have s''-wpF-ispecF-p'-B: ⋀ p' B. (p',B) ∈ set (succsF Π (fst
s'')) ⟹ Π,s'' ⊨  B ⌊⇛⌋
                       (wpF Π (fst s'') p' (⌊⋀⌋ [safeF Π p',ispecF Π p' ((d − l) − 1)]))
                        proof (intro semImpI wf-Pi)
                          fix p''' B'
                          assume p'''-B'-succsF: (p''',B') ∈ set (succsF Π (fst s''))
                          assume s''-B': Π,s'' ⊨ B'
                          show Π,s'' ⊨ wpF Π (fst s'') p''' ⌊⋀⌋[safeF Π p''', ispecF Π
p''' (d − l − 1)] (is ?wpG)
                            proof −
                              obtain p'' m'' where s''-def: s'' = (p'',m'')
                                by fastsimp

                              from wf-Pi s''-B' p'''-B'-succsF s''-def
                                obtain m''' where s''-s'''-effS: ((p'',m''),(p''',m''')) ∈
(effS Π)
                                apply −
                                apply (drule succsFprogress)
                                apply simp+
                                apply fastsimp
                                done

                              obtain s''' where s'''-def: s''' = (p''',m''')
                                by fastsimp

                              from s''-def s'''-def s''-s'''-effS s''-B' p'''-B'-succsF
                              have s''-s'''-effSB: (s'',s''') ∈ (effS_B Π)
                                apply −
                                apply (simp add: effS_B-def branch-def)
                                apply fastsimp
                                done

                              from s''-ReachableFromIn-l s''-s'''-effSB
                            have s'''-ReachableFromIn-l': s''' ∈ ReachableFromIn (effS_B
```

Π) {$s$} ($l$+1)

**apply** −
**apply** (*rule k-step*)
**apply** *simp*+
**done**

**from** $s'''$-*ReachableFromIn-l'*
**have** $s'''$-*ReachableFrom-s*: $s''' \in$ *ReachableFrom* (*effS*$_B$ Π)
{$s$}
**by** (*rule ReachableFromIn-ReachableFrom*)

**from** $s'''$-*ReachableFrom-s s-ReachableFrom*
**have** $s'''$-*ReachableFrom*: $s''' \in$ *ReachableFrom* (*effS*$_B$ Π)
{$s$. Π,$s \models$ *initF* Π}
**apply** −
**apply** (*rule ReachableFrom-trans*)
**apply** *simp*
**apply** *simp*
**done**

**from** $s'''$-*ReachableFrom isSafe*
**have** $s'''$-*safeF*: Π,$s''' \models$ *safeF* Π (*fst* $s'''$)
**by** *fastsimp*

**from** *hyp* $s'''$-*ReachableFromIn-l'* *Suc n-d-l*
**have** $s'''$-*ispecF*: Π,$s''' \models$ *ispecF* Π (*fst* $s'''$) ($d$ − ($l$+1))
**apply** −
**apply** (*erule-tac* $x$=$d$ − ($l$+1) **in** *allE*)
**apply** *simp*
**apply** (*subgoal-tac* $d$ − *Suc* $l$ < $d$ − $l$)
**prefer** 2
**apply** *arith*
**apply** *simp*
**done**

**from** $s'''$-*ispecF* $s'''$-*safeF* $s'''$-*def*
**have** $s'''$-*safeF-ispecF*: Π, $s''' \models \bigwedge$ [*safeF* Π $p'''$,*ispecF* Π
$p'''$ ($d$ − $l$ − 1)]
**apply** −
**apply** (*simp add*: *semConj*)
**done**

**from** *wf-Pi* $p'''$-$B'$-*succsF* $s''$-$B'$ $s''$-$s'''$-*effS* $s'''$-*safeF-ispecF*
$s''$-*def* $s'''$-*def*

```
                show ?wpG
                  apply −
                  apply (simp only: fst-conv snd-conv)
                  apply (rule completeWpF)
                  apply simp+
                  done
              qed
            qed


            from s''-wpF-ispecF-p'-B
            show ?thesis
              apply −
              apply (simp add: semConj split-def)
              done
          qed

          from Suc s''-wpF-ispecF specF-s''
          show ?G2''
            apply −
            apply (simp add: ispecF-def)
            done
        qed
      qed
    qed
  qed

  have s-ReachableFromIn-s: s ∈ ReachableFromIn (effS_B Π) {s} 0
    apply −
    apply (rule k-init)
    by simp



  from s-ReachableFromIn-s ispecF-trans[where k=d]
  have s-ispecF: Π,s ⊨ ispecF Π (fst s) d
    apply −
    apply (erule-tac x=0 in allE)
    apply simp
    done

  from s-safeF s-ispecF anF-ispecF
  show ?G2'
    by (simp add: aF-def)
qed
```

```
        qed
      qed
    qed
  qed
qed

  show ?G1
  proof (intro ballI impI)
    fix s
    assume s-ReachableFrom: s ∈ ReachableFrom (effS_B Π) (Starters Π)
    show Π,s ⊨ aF Π (fst s) ∧ Π,s ⊨ safeF Π (fst s) (is ?G1')
      proof −
        from s-ReachableFrom
        obtain r where s-ReachableFromIn: s ∈ ReachableFromIn (effS_B Π) (Starters Π) r
          apply −
          apply (drule ReachableFrom-ReachableFromIn)
          apply fastsimp
          done

        from G2 s-ReachableFromIn
        show ?G1'
          by fastsimp
      qed
    qed
  qed

from G1 show strongAn Π
  by (simp add: strongAn-def)
qed



end
```