

---

# Hyperproperties: Verification of Proofs

---

Denis L. Bueno   Michael R. Clarkson

{dlb335,clarkson}@cs.cornell.edu  
Department of Computer Science  
Cornell University

Computing and Information Science Technical Report  
<http://hdl.handle.net/1813/11153>

July 25, 2008

# Hyperproperties: Verification of Proofs\*

Denis L. Bueno   Michael R. Clarkson  
{dlb335,clarkson}@cs.cornell.edu  
Department of Computer Science  
Cornell University

July 25, 2008

## Abstract

This paper formalizes some proofs by Clarkson and Schneider about hyperproperties. The proofs are mechanically verified using the proof assistant Isabelle.

## 1 Introduction

Properties are sets of execution traces, and hyperproperties are sets of properties. This paper formalizes Clarkson and Schneider’s theory of hyperproperties [3] using Isabelle/HOL [4]. We present human-readable, mechanically-verified proofs of the propositions and theorems in [3]—except those related to topology, which we leave for future work. The proofs given here are formal analogues of informal proofs that were given in a previous technical report [2]. Thus, in addition to verifying the propositions and theorems, we have also verified the original proofs themselves.

This document was produced from L<sup>A</sup>T<sub>E</sub>X output, which was generated from Isabelle theory files. Those theory files are available for download from the same URL that hosts this technical report [1]. The numbering of propositions and theorems in this document follows the numbering in [2, 3].

---

\*Supported in part by AFOSR grant F9550-06-0019, National Science Foundation Grants 0430161 and CCF-0424422 (TRUST), and a gift from Microsoft Corporation. Denis Bueno is supported by a Sandia National Laboratories Fellowship; Michael Clarkson is supported by an Intel Foundation PhD Fellowship.

```

theory HyperDefs
imports Main LList2 LaTeXsugar OptionalSugar
begin

```

```

notation {} ( $\emptyset$ )

```

## 2 Definitions

```

typedecl state
— An abstract notion of a state.

```

```

types trace = state llist
— Traces are (possibly infinite) lists of states.

```

```

consts States :: state set ( $\Sigma$ )
— An abstract set of states.

```

```

consts BottomState :: state
syntax (latex)
BottomState :: state ( $\perp$ )

```

```

consts DummyState :: state

```

We assume the existence of one `DummyState`, which is used by Theorem 3 and Proposition 3.

```

axioms DummyState-is-State: DummyState  $\in$   $\Sigma$ 

```

```

constdefs
psi-fin :: trace set ( $\Psi_{\text{fin}}$ )
 $\Psi_{\text{fin}} \triangleq \Sigma^*$ 
psi-inf :: trace set ( $\Psi_{\text{inf}}$ )
 $\Psi_{\text{inf}} \triangleq \Sigma^\omega$ 
 $\Psi :: \text{trace set}$ 
 $\Psi \triangleq \Psi_{\text{fin}} \cup \Psi_{\text{inf}}$ 

```

```

types
property = trace set
hyperproperty = property set

```

```

constdefs
Prop :: property set

```

$Prop \triangleq Pow \Psi_{\text{inf}}$

$HP :: \text{hyperproperty set}$

$HP \triangleq Pow Prop$

### consts

$\text{property-satisfies} :: \text{trace set} \Rightarrow \text{property} \Rightarrow \text{bool} ((- \models -) [80,80] 80)$

$\text{hyperproperty-satisfies} :: \text{trace set} \Rightarrow \text{hyperproperty} \Rightarrow \text{bool} ((- \models -) [80,80] 80)$

### defs (overloaded)

$\text{property-satisfies-def}: ts \models p \triangleq ts \subseteq p$

$\text{hyperproperty-satisfies-def}: ts \models h \triangleq ts \in h$

### constdefs

$\text{property-lift} :: \text{property} \Rightarrow \text{hyperproperty} ([[ - ]] 80)$

$\text{property-lift } p \triangleq Pow p$

**notation**  $\text{property-lift} ([ - ] 80)$

### constdefs

$\text{trace-set-prefix} :: \text{trace set} \Rightarrow \text{trace set} \Rightarrow \text{bool} (\text{infix } \leq 80)$

$\text{trace-set-prefix-def}:$

$T \leq T' \triangleq \forall t. t \in T \longrightarrow (\exists t'. t' \in T' \wedge t \leq t')$

$Obs :: \text{trace set set}$

$Obs \triangleq \{ts. ts \subseteq \Psi_{\text{fin}} \wedge \text{finite } ts\}$

$sp :: \text{property} \Rightarrow \text{bool}$

$sp P \triangleq P \in Prop \wedge$

$(\forall t \in \Psi_{\text{inf}}. t \notin P \longrightarrow$

$(\exists m \in \Psi_{\text{fin}}. m \leq t \wedge$

$(\forall t' \in \Psi_{\text{inf}}. m \leq t' \longrightarrow t' \notin P)))$

$SP :: \text{property set}$

$SP \triangleq \{P. sp P\}$

$\text{false-p} :: \text{property}$

$\text{false-p} \triangleq \emptyset$

$\text{shp} :: \text{hyperproperty} \Rightarrow \text{bool}$

$\text{shp } H \triangleq H \in HP \wedge$

$(\forall T \in Prop. T \notin H \longrightarrow$

$(\exists M \in Obs. M \leq T \wedge$

$(\forall T' \in Prop. M \leq T' \longrightarrow T' \notin H)))$

$SHP :: \text{hyperproperty set}$

$SHP \triangleq \{hp. shp hp\}$

$\text{false-hp} :: \text{hyperproperty}$

$false\text{-}hp \triangleq [false\text{-}p]$

$lp :: property \Rightarrow bool$

$lp L \triangleq L \in Prop \wedge (\forall t \in \Psi_{fin}. (\exists t' \in \Psi_{inf}. t \leq t' \wedge t' \in L))$

$LP :: property\ set$

$LP \triangleq \{P. lp P\}$

$lhp :: hyperproperty \Rightarrow bool$

$lhp H \triangleq H \in HP \wedge (\forall T \in Obs. (\exists T' \in Prop. T \leq T' \wedge T' \in H))$

$LHP :: hyperproperty\ set$

$LHP \triangleq \{hp . lhp hp\}$

$true\text{-}Prop :: property$

$true\text{-}Prop \triangleq \Psi_{inf}$

$true\text{-}HP :: hyperproperty$

$true\text{-}HP \triangleq Prop$

end

theory *Hyper*  
imports *HyperDefs*  
begin

### 3 Proposition 1

#### 3.1 Lemmas

lemma *property-lifts-into-hyperproperty*:

assumes  $P\text{-}Prop: P \in Prop$

shows  $[P] \in HP$

using  $P\text{-}Prop$

unfolding *property-lift-def Prop-def HP-def* by blast

#### 3.2 Proposition

theorem *proposition-1-oif*:

assumes  $S\text{-}Prop: S \in Prop$  and  $S\text{-}SP: S \in SP$

shows  $[S] \in SHP$

proof –

have *lift-S-HP*:  $[S] \in HP$

using  $S\text{-}Prop$  *property-lifts-into-hyperproperty* by blast

{

```

fix  $T :: \text{property}$ 
assume  $T\text{-st}: T \in \text{Prop } T \notin [S]$ 
from  $\langle T \notin [S] \rangle$  have  $\neg(T \subseteq S)$  by (simp add: property-lift-def)
then obtain  $t$  where  $t\text{-st}: t \in T \ t \notin S$  by blast

have  $\exists m. m \in \Psi_{\text{fin}} \wedge m \leq t \wedge (\forall t' \in \Psi_{\text{inf}}. m \leq t' \longrightarrow t' \notin S)$ 
proof –
  from  $t\text{-st}$  and  $T\text{-st}$  have  $t\text{-psi-inf}: t \in \Psi_{\text{inf}}$ 
  unfolding Prop-def by blast
  with  $S\text{-Prop}$  and  $S\text{-SP}$  and  $T\text{-st}$  and  $t\text{-st}$ 
  show ?thesis unfolding SP-def Prop-def sp-def by blast
qed
then obtain  $m$  where  $m\text{-st}: m \in \Psi_{\text{fin}} \ m \leq t \ \forall t'. t' \in \Psi_{\text{inf}} \wedge m \leq t' \longrightarrow$ 
 $t' \notin S$ 
  by blast

let  $?M = \{m\}$ 
from  $m\text{-st}$  and  $t\text{-st}$  have  $M\text{-prf-}T: ?M \leq T$ 
  unfolding trace-set-prefix-def by blast
with  $m\text{-st}$  and  $t\text{-st}$  have  $M\text{-Obs}: ?M \in \text{Obs}$ 
  unfolding Obs-def by blast

{
  fix  $T' :: \text{property}$ 
  assume  $T'\text{-st}: T' \in \text{Prop } ?M \leq T'$ 

  then have  $\exists t' \in T'. m \leq t'$ 
    by (simp only: trace-set-prefix-def) blast
  then obtain  $t'$  where  $t'\text{-st}: t' \in T' \ m \leq t' ..$ 
  with  $m\text{-st}$  and  $T'\text{-st}$  have  $t'\text{-out-}S: t' \notin S$ 
    unfolding Prop-def by blast
  from  $T'\text{-st}$  and  $S\text{-Prop}$  and  $S\text{-SP}$  and  $t'\text{-st}$  and  $t'\text{-out-}S$ 
  have  $T' \notin [S]$  unfolding property-lift-def by blast
}
hence  $\forall T'. T' \in \text{Prop} \wedge ?M \leq T' \longrightarrow T' \notin [S]$  by blast

with  $m\text{-st}$  and  $M\text{-prf-}T$  and  $M\text{-Obs}$ 
have  $\exists M. M \in \text{Obs} \wedge M \leq T \wedge (\forall T'. T' \in \text{Prop} \wedge M \leq T' \longrightarrow T' \notin [S])$ 
  by blast
}
thus ?thesis using lift-S-HP unfolding SHP-def shp-def by blast
qed

```

**lemma** *prefix-set-has-longest*:

```

fixes  $t :: 'a$  llist
assumes  $X$ -fin: finite  $X$  and  $X$ -non-empty:  $X \neq \emptyset$ 
and  $X$ -prefix- $t$ :  $\forall x \in X. x \leq t$ 
shows  $\exists m \in X. (\forall x \in X. x \leq m)$ 
using prems
proof (induct  $X$  rule: Finite-Set.finite-ne-induct)
  fix  $x :: 'a$  llist show  $\exists m \in \{x\}. \forall x \in \{x\}. x \leq m$  by blast
next
  fix  $x :: 'a$  llist and  $F :: 'a$  llist set
  assume
     $R: \forall x \in F. x \leq t \implies \exists m \in F. \forall x \in F. x \leq m$ 
    and  $t$ -upper-bound:  $\forall x \in \text{insert } x F. x \leq t$ 
  then obtain  $m$  where
     $m$ -in- $F$ :  $m \in F$  and  $m$ -le- $t$ :  $m \leq t$  and  $x$ -le- $t$ :  $x \leq t$ 
    and  $m$ -max- $F$ :  $\forall x \in F. x \leq m$  using  $R$  by (auto dest:  $R$ )
  from  $m$ -le- $t$   $x$ -le- $t$  have  $m \leq x \vee x \leq m$  by (rule pref-locally-linear)
  thus  $\exists m \in \text{insert } x F. \forall x \in \text{insert } x F. x \leq m$ 
proof
  assume  $m \leq x$  with  $m$ -max- $F$ 
  have  $\forall xa \in \text{insert } x F. xa \leq x$  by auto
  thus ?thesis by blast
next assume  $x \leq m$  with  $m$ -max- $F$ 
  have  $\forall xa \in \text{insert } x F. xa \leq m$  by auto
  thus ?thesis using  $m$ -in- $F$  by blast
qed
qed

```

```

theorem proposition-1-if:
  assumes  $S$ -Prop:  $S \in \text{Prop}$  and lift- $S$ -shp:  $[S] \in \text{SHP}$ 
  shows  $S \in \text{SP}$ 
proof –
  {
    — Show that  $t$  has finite bad thing  $m$ .
    fix  $t :: \text{trace}$ 
    assume  $t$ -st:  $t \notin S \{t\} \in \text{Prop}$ 
    then have  $t$ -out-lift- $S$ :  $\{t\} \notin [S]$  by (simp add: property-lift-def)

    obtain  $M$  where
       $M$ -st:  $M \in \text{Obs}$   $M \leq \{t\} \vee T'. T' \in \text{Prop} \wedge M \leq T' \implies T' \notin [S]$ 
      using  $t$ -out-lift- $S$  and  $t$ -st and  $S$ -Prop and lift- $S$ -shp
      unfolding SHP-def shp-def
      by blast

    have  $\exists ms \in \Psi_{\text{fin}}. ms \in M \wedge ms \leq t \wedge (\forall m \in M. m \leq ms)$ 

```

```

proof –
  have  $M\text{-pfx-}t: \forall m \in M. m \leq t$ 
    using  $M\text{-st}$  unfolding  $\text{trace-set-prefix-def}$  by  $\text{blast}$ 
  have  $M\text{-nonempty}: M \neq \emptyset$ 
  proof ( $\text{rule ccontr}$ )
  {
    assume  $M\text{-empty}: \neg M \neq \emptyset$ 
    {
      fix  $T' :: \text{property}$  assume  $T' \in \text{Prop}$ 
      with  $M\text{-empty}$  have  $M \leq T'$  unfolding  $\text{trace-set-prefix-def}$  by  $\text{blast}$ 
    }
    hence  $M\text{-pfx-Prop}: \forall T' \in \text{Prop}. M \leq T'$  by  $\text{blast}$ 
    have  $\emptyset \in \text{Prop}$  unfolding  $\text{Prop-def}$  by  $\text{blast}$ 
    hence  $M \leq \emptyset$  using  $M\text{-pfx-Prop}$  by  $\text{blast}$ 
    hence  $\emptyset \notin [S]$  using  $M\text{-st}$  and  $(\emptyset \in \text{Prop})$  by  $\text{blast}$ 
    have  $\emptyset \in [S]$  using  $\text{property-lift-def}$  by  $\text{blast}$ 
    from  $(\emptyset \in [S])$  and  $(\neg \emptyset \in [S])$  have  $\text{False}$  by  $\text{blast}$ 
  }
  thus  $\neg M \neq \emptyset \implies \text{False}$  by  $\text{blast}$ 
qed

```

```

  have  $M\text{-fin}: \text{finite } M$  using  $M\text{-st}$  unfolding  $\text{Obs-def}$  by  $\text{blast}$ 
  from this obtain  $ms$  where  $ms\text{-st}: ms \in M \forall x \in M. x \leq ms$ 
    using  $M\text{-pfx-}t$  and  $M\text{-nonempty}$ 
    apply ( $\text{insert prefix-set-has-longest [where } t=t \text{ and } X=M], \text{blast}$ )
    done
  hence  $ms\text{-psi-fin}: ms \in \Psi_{\text{fin}}$  using  $M\text{-st}$  unfolding  $\text{Obs-def}$  by  $\text{blast}$ 
  have  $ms\text{-pfx-}t: ms \leq t$  using  $ms\text{-st}$  and  $M\text{-st}$  unfolding  $\text{trace-set-prefix-def}$ 
    by  $\text{blast}$ 
  from  $ms\text{-psi-fin}$  and  $ms\text{-st}$  and  $ms\text{-pfx-}t$ 
  show  $\exists ms \in \Psi_{\text{fin}}. ms \in M \wedge ms \leq t \wedge (\forall m \in M. m \leq ms)$ 
    by  $\text{blast}$ 
qed
from this obtain  $m\text{-star}$  where
   $m\text{-star-st}: m\text{-star} \in \Psi_{\text{fin}} \quad m\text{-star} \in M \quad m\text{-star} \leq t$ 
   $\forall m \in M. m \leq m\text{-star}$ 
by  $\text{auto}$ 
{
  fix  $t'$ 
  assume  $t'\text{-st}: \{t'\} \in \text{Prop} \quad m\text{-star} \leq t'$ 
  let  $?T' = \{t'\}$ 
  have  $M \leq ?T'$ 
  proof –

```

```

    {
      fix m
      assume m ∈ M
      with m-star-st have m ≤ m-star by blast
      with t'-st have m ≤ t' using llist-le-trans by blast
    }
    thus M ≤ ?T' unfolding trace-set-prefix-def by blast
  qed

  with M-st and t'-st have ?T' ∉ [S] by blast
  hence t' ∉ S unfolding property-lift-def by blast
}

with m-star-st have ∃ m ∈ Ψfin. m ≤ t ∧ (∀ t' ∈ Ψinf. m ≤ t' → t' ∉ S)
  unfolding Prop-def
  by blast
}
thus ?thesis
  using S-Prop
  unfolding SP-def sp-def Prop-def by blast
qed

```

## 4 Proposition 2

**theorem** *proposition-2-oif*:

fixes  $L :: \text{trace set}$   
 assumes  $L\text{-Prop}: L \in \text{Prop}$  and  $L\text{-LP}: L \in \text{LP}$   
 shows  $[L] \in \text{LHP}$

**proof** –

have  $\text{lift-}L\text{-HP}: [L] \in \text{HP}$   
 using  $L\text{-Prop}$  *property-lifts-into-hyperproperty* by blast

```

{
  fix M assume M-st: M ∈ Obs
  {
    fix m assume m-st: m ∈ M
    have ∃ t. m ≤ t ∧ t ∈ L
    proof –
      from m-st and M-st have m ∈ Ψfin
      unfolding Obs-def by blast
      with L-Prop and L-LP and m-st show ?thesis
      unfolding LP-def lp-def Prop-def by blast
    qed
  }
  hence M-more: ∀ m ∈ M. (∃ t. m ≤ t ∧ t ∈ L) by blast
}

```

```

let ?T = {tm. ∃ m ∈ M. m ≤ tm ∧ tm ∈ L}
have ?T ⊆ L by blast
hence T-in-lift: ?T ∈ [L] unfolding property-lift-def by blast
with M-more have M-px-T: M ≤ ?T
  unfolding trace-set-prefix-def by blast
have ?T ∈ Prop using M-st L-Prop
  unfolding Prop-def psi-inf-def Obs-def psi-fin-def
  by blast
with T-in-lift and M-px-T and L-Prop
  have ∃ T. T ∈ Prop ∧ M ≤ T ∧ T ∈ [L] by blast
}
thus [L] ∈ LHP using lift-L-HP unfolding LHP-def lhp-def by blast
qed

```

**theorem** *proposition-2-if:*

**fixes**  $L :: \text{trace set}$

**assumes**  $L\text{-Prop}: L \in \text{Prop}$  **and**  $L\text{-lift-lhp}: [L] \in \text{LHP}$

**shows**  $L \in \text{LP}$

**proof** –

{ **fix**  $t :: \text{trace}$  **assume**  $t\text{-st}: t \in \Psi_{\text{fin}}$

**let**  $?T = \{t\}$

**obtain**  $T'$  **where**  $T'\text{-st}: ?T \leq T'$   $T' \in [L]$   $T' \in \text{Prop}$

**proof** –

**from**  $t\text{-st}$  **have**  $t\text{-Obs}: \{t\} \in \text{Obs}$  **using**  $\text{Obs-def}$  **by** *blast*

**hence**  $\exists T' \in \text{Prop}. ?T \leq T' \wedge T' \in [L]$

**using**  $L\text{-lift-lhp}$  **unfolding**  $LHP\text{-def}$   $lhp\text{-def}$  **by** *blast*

**thus** *thesis* **by** *auto*

**qed**

**then obtain**  $t'$  **where**  $t'\text{-st}: t \leq t'$   $t' \in T'$   $t' \in \Psi_{\text{inf}}$

**unfolding**  $\text{trace-set-prefix-def}$   $\text{Prop-def}$  **by** *blast*

**have**  $t' \in L$  **using**  $\langle t' \in T' \rangle$  **and**  $\langle T' \in [L] \rangle$

**unfolding**  $\text{property-lift-def}$  **by** *blast*

**with**  $t'\text{-st}$  **have**  $\exists t' \in \Psi_{\text{inf}}. t \leq t' \wedge t' \in L$  **by** *blast*

}

**thus**  $L \in \text{LP}$  **unfolding**  $LP\text{-def}$   $lp\text{-def}$  **using**  $L\text{-Prop}$  **by** *blast*

**qed**

## 5 Theorem 3

### 5.1 Definitions and Lemmas

**constdefs**

*Safe* :: *hyperproperty*  $\Rightarrow$  *hyperproperty*

$Safe\ P \triangleq \{T \in Prop. (\forall M \in Obs. M \leq T \longrightarrow (\exists T' \in Prop. M \leq T' \wedge T' \in P))\}$   
 $Live \ ::\ hyperproperty \Rightarrow hyperproperty$   
 $Live\ P \triangleq P \cup (Prop - Safe\ P)$

**lemma** *Safe-is-HP*:  
**fixes**  $P \ ::\ hyperproperty$   
**assumes**  $P \in HP$   
**shows**  $Safe\ P \in HP$   
**unfolding** *Safe-def HP-def* **by** *blast*

**lemma** *Live-is-HP*:  
**fixes**  $P \ ::\ hyperproperty$   
**assumes**  $P-HP: P \in HP$   
**shows**  $Live\ P \in HP$   
**using**  $P-HP$   
**unfolding** *Live-def HP-def* **by** *blast*

**lemma** *Safe-is-hypersafety*:  
**fixes**  $P \ ::\ hyperproperty$   
**assumes**  $P-HP: P \in HP$   
**shows**  $Safe\ P \in SHP$   
**using**  $P-HP\ Safe-is-HP$   
**unfolding** *Safe-def SHP-def shp-def*  
**by** *blast*

**lemma** *P-subset-Safe-P*:  
**fixes**  $P \ ::\ hyperproperty$   
**assumes**  $P-HP: P \in HP$   
**shows**  $P \subseteq Safe\ P$   
**using**  $P-HP$   
**unfolding** *Safe-def HP-def*  
**by** *blast*

**lemma** *stutter-append-is-infinite*:  
**fixes**  $x \ ::\ trace$   
**assumes**  $x-fin: x \in \Psi_{fin}$  **and**  $s-st: s \in \Sigma$   
**shows**  $(x @@ lconst\ s) \in \Psi_{inf}$   
**proof** –  
**from**  $s-st$  **have**  $lconst\ s \in inflsts\ \Sigma$   
**by** *(rule lconstT [of s Σ])*  
**thus**  $(x @@ lconst\ s) \in \Psi_{inf}$   
**using**  $x-fin\ s-st\ lapp-fin-infT$   
**unfolding** *psi-fin-def psi-inf-def*  
**by** *blast*

qed

**constdefs**

*asInfinite* :: trace  $\Rightarrow$  trace

*asInfinite*  $t \triangleq$  if  $LNil = t$  then *lconst DummyState* else  $t @@ (lconst (llast t))$

— Converts a finite trace to an infinite trace. If the given finite trace is non-empty, it returns a suffix in which the final state is infinitely stuttered; otherwise it returns the constant *DummyState* trace.

**lemma** *llast-in-trace-alphabet*:

**assumes**  $t \in \Psi_{\text{fin}}$

**shows**  $t \neq LNil \longrightarrow llast t \in \Sigma$  (**is** ?*P*  $t$ )

**using** *prems*

**unfolding** *psi-fin-def*

**by** (*induct t rule: finlsts.induct*) *auto*

**lemma** *asInfinite-correctness*:

**assumes** *t-fin*:  $t \in \Psi_{\text{fin}}$

**shows**  $asInfinite t \in \Psi_{\text{inf}} \wedge t \leq asInfinite t$

**proof** *cases*

**assume**  $LNil = t$

**thus** ?*thesis* **unfolding** *asInfinite-def psi-inf-def* **using** *DummyState-is-State*

**by** (*simp add: lconstT [of DummyState  $\Sigma$ ]*)

**next**

**assume** *t-positive*:  $LNil \neq t$

**with** *t-fin* **have** *res-inf*:  $asInfinite t \in \Psi_{\text{inf}}$

**proof**—

**have**  $llast t \in \Sigma$  **using** *t-positive t-fin llast-in-trace-alphabet* **by** *simp*

**moreover**

**have**  $lconst (llast t) \in \Psi_{\text{inf}}$

**using** *t-fin t-positive*  $\langle llast t \in \Sigma \rangle$  **unfolding** *psi-fin-def psi-inf-def*

**by** (*simp add: lconstT [of llast t  $\Sigma$ ]*)

**moreover**

**have**  $t @@ lconst (llast t) \in \Psi_{\text{inf}}$

**using** *t-fin*  $\langle llast t \in \Sigma \rangle$

**by** (*simp add: stutter-append-is-infinite [of t llast t]*)

**ultimately**

**show**  $asInfinite t \in \Psi_{\text{inf}}$  **unfolding** *asInfinite-def*

**using** *t-positive* **by** *simp*

**qed**

**from** *t-fin* **and** *t-positive*

**have**  $t \leq asInfinite t$

**unfolding** *psi-fin-def asInfinite-def* **using** *le-lappend* **by** *simp*

**with** *res-inf* **show** ?*thesis* ..

qed

**lemma** *Live-is-hyperliveness*:

**fixes**  $P::\text{hyperproperty}$

**assumes**  $P\text{-HP}: P \in \text{HP}$

**shows**  $\text{Live } P \in \text{LHP}$

**proof** –

**have**  $\text{Live-HP}: \text{Live } P \in \text{HP}$  **using**  $P\text{-HP}$  *Live-is-HP* **by** *blast*

{

**fix**  $T$  **assume**  $T\text{-st}: T \in \text{Obs}$

**have**  $\exists T' \in \text{Prop}. T \leq T' \wedge T' \in \text{Live } P$

**proof** *cases*

**assume**  $\exists T' \in \text{Prop}. T \leq T' \wedge T' \in P$

**then obtain**  $T'$  **where**  $T'\text{-st}: T' \in \text{Prop}$   $T \leq T'$   $T' \in P$  **by** *blast*

**hence**  $T' \in \text{Live } P$  **unfolding** *Live-def* **by** *blast*

**thus** *?thesis* **using**  $T'\text{-st}$  **by** *blast*

**next**

**assume**  $T'\text{-non-extends}: \neg(\exists T' \in \text{Prop}. T \leq T' \wedge T' \in P)$

{

**fix**  $T'$  **assume**  $T'\text{-extends-}T: T' \in \text{Prop}$   $T \leq T'$

**hence**  $T' \notin P$  **using**  $T'\text{-non-extends}$  **by** *blast*

**hence**  $T' \notin \text{Safe } P$

**proof** –

**have**  $\exists T \in \text{Obs}. T \leq T' \wedge (\forall T' \in \text{Prop}. \neg(T \leq T') \mid (T' \notin P))$

**using**  $T\text{-st}$  **and**  $T'\text{-extends-}T$  **and**  $T'\text{-non-extends}$  **by** *blast*

**hence**  $\neg(\forall M \in \text{Obs}. M \leq T' \longrightarrow$

$(\exists T'' \in \text{Prop}. M \leq T'' \wedge T'' \in P))$

**by** *blast*

**thus** *?thesis* **using**  $\langle T' \in \text{Prop} \rangle$  **unfolding** *Safe-def* **by** *blast*

**qed**

**hence**  $T' \in (\text{Prop} - \text{Safe } P)$  **using**  $\langle T' \in \text{Prop} \rangle$  **by** *blast*

}

**hence**  $\text{all-pfx}: \forall T' \in \text{Prop}. T \leq T' \longrightarrow T' \in \text{Prop} - \text{Safe } P$  **by** *simp*

**show**  $\exists T' \in \text{Prop}. T \leq T' \wedge T' \in \text{Live } P$

**proof** –

**let**  $?T' = \{\text{asInfinite } x \mid x. x \in T\}$

**have**  $T'\text{-suff}: T \leq ?T'$  **using**  $\text{asInfinite-correctness}$   $T\text{-st}$

**unfolding**  $\text{trace-set-prefix-def}$   $\text{Obs-def}$  **by** *blast*

**have**  $T'\text{-Prop}: ?T' \in \text{Prop}$  **using**  $T\text{-st}$   $\text{asInfinite-correctness}$

**unfolding**  $\text{Obs-def}$   $\text{Prop-def}$  **by** *blast*

**from**  $T'\text{-suff}$  **and**  $T'\text{-Prop}$  **have**  $?T' \in \text{Prop} - \text{Safe } P$  **using**  $\text{all-pfx}$  **by**

*blast*

**with**  $T'\text{-suff}$  **and**  $T'\text{-Prop}$  **show** *?thesis* **unfolding** *Live-def* **by** *blast*

**qed**

```

    qed
  }
  thus ?thesis using Live-HP unfolding Live-def LHP-def lhp-def by blast
qed

```

## 5.2 Theorem

**theorem** *theorem-3*:

**fixes**  $P :: \text{trace set set}$

**assumes**  $P\text{-HP}: P \in \text{HP}$

**shows**  $\exists S \in \text{SHP}. \exists L \in \text{LHP}. P = S \cap L$

**proof** –

**let**  $?S = \text{Safe } P$  **let**  $?L = \text{Live } P$

**have**  $?S \cap ?L = (P \cup \text{Safe } P) \cap (P \cup (\text{Prop} - \text{Safe } P))$

**unfolding** *Live-def* **using**  $P\text{-HP}$   $P\text{-subset-Safe-}P$  **by** *blast*

**also have**  $(P \cup \text{Safe } P) \cap (P \cup (\text{Prop} - \text{Safe } P))$

$= P \cap (\text{Safe } P \cup (\text{Prop} - \text{Safe } P))$

**using**  $P\text{-HP}$  **unfolding** *HP-def* **by** *blast*

**also have**  $P \cap (\text{Safe } P \cup (\text{Prop} - \text{Safe } P)) = P \cap \text{Prop}$

**unfolding** *Safe-def* **by** *blast*

**also have**  $P \cap \text{Prop} = P$  **using**  $P\text{-HP}$  **unfolding** *HP-def* **by** *blast*

**finally have** *witness*:  $?S \cap ?L = P$  **by** *blast*

**have** *Safe-SHP*:  $\text{Safe } P \in \text{SHP}$  **using** *Safe-is-hypersafety*  $P\text{-HP}$  **by** *blast*

**have** *Live-LHP*:  $\text{Live } P \in \text{LHP}$  **using** *Live-is-hyperliveness*  $P\text{-HP}$  **by** *blast*

**show** ?thesis **using** *Safe-SHP* *Live-LHP* *witness* **by** *blast*

**qed**

## 6 Theorem 1

### 6.1 Definitions and Lemmas

**constdefs**

*Systems*  $:: \text{trace set set}$

*Systems*  $\triangleq \{ts. ts \neq \emptyset \wedge ts \subseteq \Psi_{\text{inf}}\}$

*refinedby*  $:: \text{trace set} \Rightarrow \text{trace set} \Rightarrow \text{bool}$  (**infix**  $\leq 80$ )

$S \leq S' \triangleq S' \subseteq S$

*rc*  $:: \text{hyperproperty} \Rightarrow \text{bool}$

$rc\ H \triangleq \forall S \in \text{Systems}. S \models H \longrightarrow$

$(\forall S' \in \text{Systems}. S \leq S' \longrightarrow S' \models H)$

*RC*  $:: \text{hyperproperty set}$

*RC*  $\triangleq \{H \in \text{HP}. rc\ H\}$

**axioms** *safety-and-liveness-only-if-true*:

$\llbracket p \in LP; p \in SP \rrbracket \implies p = \text{true-Prop}$

— Any property which is both safety and liveness is the *true* property. This is axiomatised since it is well-known about the theory of properties.

**lemma** *hypersafety-and-hyperliveness-onlyif-true*:

**fixes**  $H :: \text{hyperproperty}$

**assumes**  $H\text{-SHP}: H \in \text{SHP}$  **and**  $H\text{-LHP}: H \in \text{LHP}$

**shows**  $H = \text{true-HP}$

**proof** (*rule ccontr*)

**have**  $H\text{-HP}: H \in \text{HP}$  **using**  $H\text{-SHP}$  **unfolding**  $\text{SHP-def shp-def}$  **by** *blast*

{

**assume**  $H\text{-untrue}: H \neq \text{true-HP}$

**then obtain**  $T\text{star}$  **where**  $T\text{star-st}: T\text{star} \in \text{Prop}$   $T\text{star} \notin H$

**using**  $H\text{-HP}$  **unfolding**  $\text{HP-def true-HP-def Prop-def}$  **by** *blast*

**obtain**  $M$  **where**  $M\text{-st}: M \in \text{Obs}$   $M \leq T\text{star}$

$\forall T' \in \text{Prop}. M \leq T' \implies T' \notin H$

**using**  $H\text{-SHP}$   $T\text{star-st}$

**unfolding**  $\text{SHP-def shp-def}$  **by** *blast*

**then obtain**  $Th$  **where**  $Th\text{-st}: Th \in \text{Prop}$   $M \leq Th$   $Th \in H$

**using**  $H\text{-LHP}$

**unfolding**  $\text{LHP-def lhp-def}$  **by** *blast*

**hence**  $Th \notin H$  **using**  $\langle Th \in \text{Prop} \rangle M\text{-st}$  **by** *blast*

**thus** *False* **using**  $Th\text{-st}$  **by** *blast*

}

**qed**

**lemma** *hypersafety-and-hyperliveness-onlyif-true-contrapos*:

**fixes**  $H :: \text{hyperproperty}$

**shows**  $H \neq \text{true-HP} \implies (H \notin \text{LHP} \mid H \notin \text{SHP})$

**apply** (*insert hypersafety-and-hyperliveness-onlyif-true [of H]*)

**by** *blast*

**axioms**  $\text{Ex-nontrue-Prop}: \exists l \in \text{LP}. l \neq \text{true-Prop}$

— There is a liveness property other than *true*. This is axiomatised since it is well-known about the theory of properties.

**lemma** *system-is-property*:

**fixes**  $s :: \text{trace set}$

**assumes**  $s\text{-Sys}: s \in \text{Systems}$

**shows**  $s \in \text{Prop}$

**using**  $s\text{-Sys}$

**unfolding**  $\text{Systems-def Prop-def}$  **by** *blast*

**lemma**  $\text{HP-contains-SHP}: \text{SHP} \subseteq \text{HP}$  **unfolding**  $\text{SHP-def shp-def}$  **by** *blast*

## 6.2 Theorem

**theorem** *theorem-1-relaxed*:

**shows**  $SHP \subseteq RC$

**proof** (*rule ccontr*)

**assume**  $\neg SHP \subseteq RC$

**then obtain**  $S$  **where**  $S\text{-SHP}$ :  $S \in SHP$  **and**  $S\text{-not-RC}$ :  $S \notin RC$  **by** *blast*

**have**  $S\text{-HP}$ :  $S \in HP$  **using**  $S\text{-SHP}$   $HP\text{-contains-SHP}$  **by** *blast*

**from**  $S\text{-HP}$  **and**  $S\text{-not-RC}$

**obtain**  $T T'$  **where**  $T\text{-st}$ :  $T \in Prop$   $T \in S$

**and**  $T'\text{-st}$ :  $T' \in Prop$   $T' \notin S$

**and**  $T\text{-gt-}T'$ :  $T \supseteq T'$

**unfolding**  $RC\text{-def}$   $rc\text{-def}$   $HP\text{-def}$   $Systems\text{-def}$   $Prop\text{-def}$

**unfolding**  $refinedby\text{-def}$   $hyperproperty\text{-satisfies}\text{-def}$

**by** *blast*

**from**  $T'\text{-st}$  **obtain**  $M$

**where**  $M\text{-st}$ :  $M \leq T'$  ( $\forall T'' \in Prop. M \leq T'' \longrightarrow T'' \notin S$ )

**using**  $S\text{-SHP}$  **unfolding**  $SHP\text{-def}$   $shp\text{-def}$  **by** *blast*

**have**  $M \leq T$

**using**  $M\text{-st}$   $T\text{-st}$   $T'\text{-st}$   $T\text{-gt-}T'$

**unfolding**  $trace\text{-set-prefix}\text{-def}$  **by** *blast*

**hence**  $T \notin S$  **using**  $T\text{-st}$   $M\text{-st}$  **by** *blast*

**thus** *False* **using**  $T\text{-st}$  **by** *blast*

**qed**

**theorem** *theorem-1*:  $SHP \subset RC$

**proof**

**show**  $SHP \subseteq RC$  **using** *theorem-1-relaxed* **by** *assumption*

**obtain**  $l$  **where**  $l\text{-LP}$ :  $l \in LP$  **and**  $l\text{-untrue}$ :  $l \neq \text{true-Prop}$

**using**  $Ex\text{-nontrue-Prop}$  **by** *blast*

**hence**  $cx\text{-RC}$ :  $[l] \in RC$

**unfolding**  $property\text{-lift}\text{-def}$   $LP\text{-def}$   $lp\text{-def}$   $RC\text{-def}$   $rc\text{-def}$   $Systems\text{-def}$

$refinedby\text{-def}$   $HP\text{-def}$   $Prop\text{-def}$   $psi\text{-inf}\text{-def}$   $psi\text{-fin}\text{-def}$

$hyperproperty\text{-satisfies}\text{-def}$

**by** *blast*

**from**  $l\text{-untrue}$  **have**  $[l] \neq \text{true-HP}$

**using**  $l\text{-LP}$

**unfolding**  $LP\text{-def}$   $lp\text{-def}$   $\text{true-Prop}\text{-def}$   $\text{true-HP}\text{-def}$   $property\text{-lift}\text{-def}$

$psi\text{-inf}\text{-def}$   $Prop\text{-def}$

**by** *blast*

**hence**  $[l] \notin SHP$

**proof** –

**have**  $l \in Prop$  **using**  $l\text{-LP}$  **unfolding**  $LP\text{-def}$   $lp\text{-def}$  **by** *blast*

**with**  $l\text{-LP}$  **have**  $[l] \in LHP$  **using** *proposition-2-oif* **by** *blast*

**thus**  $[l] \notin SHP$

```

using ⟨[l] ≠ true-HP⟩
      hypersafety-and-hyperliveness-only-if-true-contrapos by blast
qed
thus SHP ≠ RC using cx-RC by blast
qed

```

## 7 Proposition 3

### 7.1 Definitions and Lemmas

**constdefs**

```

Cls :: ('a set ⇒ 'a set) set
Cls ≜ {cl. ∀ T :: 'a set. T ⊆ cl T}

```

```

PIF :: hyperproperty set
PIF ≜ {{Cl T | T. T ∈ Prop} | Cl. Cl ∈ Cls}

```

```

lsingle :: 'a ⇒ 'a llist
lsingle x ≜ x##LNil

```

```

hasDummyState :: trace ⇒ bool
hasDummyState t ≜ ∃ t'. t'@@(lsingle DummyState) ≤ t

```

```

GS :: trace set
GS ≜ {t. t ∈ Ψinf ∧ hasDummyState t}

```

— The guaranteed service property, *GS*, contains infinite traces in which a designated state occurs. This definition generalizes *GS* from the technical report.

**axioms**

```

Cl-produces-Props: [ T ∈ Prop; Cl ∈ Cls ] ⇒ Cl T ∈ Prop

```

— This axiom is essentially a type signature on closures. It is axiomatised because although it is not mentioned in the technical report, it is required for Proposition 3.

```

EX-trace-sans-DummyState: ∃ t ∈ Ψinf. ¬hasDummyState t

```

— There is an infinite trace without a certain state (the *DummyState*, in this case). This is axiomatised because it is well-known about the theory of properties.

```

GS-liveness: lp GS

```

— The *GS* property is a liveness property. This is axiomatised since it is well-known.

**lemma** *GS-LHP*:  $[GS] \in LHP$   
**proof** –  
  **have**  $GS \in Prop$  **unfolding** *Prop-def* *GS-def* **by** *blast*  
  **thus** *?thesis* **using** *GS-liveness proposition-2-oif* **unfolding** *LP-def* **by** *blast*  
**qed**

**lemma** *trace-set-prefix-expanding'*:  
  **fixes**  $T :: trace\ set$   
  **assumes**  $T\text{-st}: T \leq T'$  **and**  $T'\text{-sub}: T' \subseteq T''$   
  **shows**  $T \leq T''$   
  **using**  $T\text{-st}$   $T'\text{-sub}$  **unfolding** *trace-set-prefix-def* **by** *blast*

## 7.2 Proposition

**theorem** *proposition-3-relaxed*:  
  **shows**  $PIF \subseteq LHP$   
**proof** –  
  {  
    **fix**  $P$  **assume**  $P \in PIF$   
    **then obtain**  $Cl\text{-}P$  **where**  $P\text{-st}: P = \{Cl\text{-}P\ T \mid T. T \in Prop\}$   
      **and**  $Cl\text{-}P\text{-closure}: Cl\text{-}P \in Cls$   
    **unfolding** *PIF-def* **by** *blast*  
    **have**  $P\text{-HP}: P \in HP$   
    **proof** –  
    {  
      **fix**  $x$  **assume**  $x \in P$   
      **then obtain**  $T$  **where**  $T\text{-st}: x = Cl\text{-}P\ T\ T \in Prop$   
      **using**  $P\text{-st}$  **by** *blast*  
      **hence**  $x \in Prop$  **using**  $Cl\text{-}P\text{-closure}$  *Cl-produces-Props* **by** *blast*  
    }  
    **thus** *?thesis* **unfolding** *HP-def* **by** *blast*  
  }  
**qed**  
  {  
    **fix**  $T$  **assume**  $T\text{-Obs}: T \in Obs$   
    **have**  $\exists T' \in Prop. T \leq T' \wedge T' \in P$   
    **proof** –  
    **let**  $?T\text{-inf} = \{asInfinite\ t \mid t. t \in T\}$   
    **let**  $?T' = Cl\text{-}P\ ?T\text{-inf}$   
    **have**  $T'\text{-suff}: T \leq ?T'$   
    **proof** –  
    **have**  $Cl\text{-}P\text{-monotonic}: \bigwedge X. X \subseteq Cl\text{-}P\ X$   
      **using**  $Cl\text{-}P\text{-closure}$  **unfolding** *Cls-def* **by** *blast*  
    **hence**  $Cl\text{-}P\text{-prop}: ?T\text{-inf} \subseteq Cl\text{-}P\ ?T\text{-inf}$  **by** *auto*  
    **have**  $T\text{-pfx-}T\text{-inf}: T \leq ?T\text{-inf}$   
    **using**  $T\text{-Obs}$  *asInfinite-correctness*  
  }

```

    unfolding Obs-def trace-set-prefix-def by blast
  with Cl-P-prop show ?thesis
    apply (insert trace-set-prefix-expanding' [OF T-pfx-T-inf Cl-P-prop])
    apply assumption
    done
  qed
  have ?T-inf ∈ Prop using T-Obs asInfinite-correctness
    unfolding Obs-def Prop-def by blast
  hence T'-P: ?T' ∈ P using P-st by blast
  have T'-Prop: ?T' ∈ Prop
    using (?T-inf ∈ Prop) Cl-P-closure Cl-produces-Props by blast
  with (?T' ∈ P) and T'-suff show ?thesis by blast qed
}
hence P ∈ LHP using P-HP unfolding LHP-def lhp-def by blast
}
thus PIF ⊆ LHP by blast
qed

```

**theorem** *proposition-3*:

shows  $PIF \subset LHP$

**proof**

show  $PIF \subseteq LHP$  using *proposition-3-relaxed* .

have *GS-lift-LHP*: [*GS*] ∈ *LHP* by (*simp add: GS-LHP*)

show  $PIF \neq LHP$

**proof** (*rule ccontr*)

{

assume  $PIF = LHP$

hence [*GS*] ∈ *PIF* using *GS-lift-LHP* by *simp*

then obtain *CL-GS*

where *CL-GS-st*: [*GS*] = {*CL-GS T* | *T*. *T* ∈ *Prop*}

and *CL-GS-Cls*: *CL-GS* ∈ *Cls* unfolding *PIF-def* by blast

obtain *t*

where *t-inftrace*: *t* ∈  $\Psi_{\text{inf}}$

and *t-no-Dummy*:  $\neg \text{hasDummyState } t$

using *EX-trace-sans-DummyState* by blast

hence *ts-Prop*: {*t*} ∈ *Prop* unfolding *Prop-def* by blast

have *t* ∈ *CL-GS* {*t*} using *CL-GS-Cls* unfolding *Cls-def* by blast

hence  $\neg (\text{CL-GS } \{t\} \models \text{GS})$

using *t-no-Dummy*

unfolding *property-satisfies-def GS-def* by blast

hence *False* using *CL-GS-st*

using *ts-Prop* unfolding *property-satisfies-def property-lift-def* by blast

}

thus  $\neg PIF \neq LHP \implies \text{False}$  by blast

qed

qed

## 8 Theorem 2

### 8.1 Definitions and Lemmas

We represent traces over the alphabet  $A^k$  as *'a llist llist* where *'a* is the type of elements of  $A$ . That is, instead of using  $k$ -tuples, we use llists of length  $k$ .

**constdefs**

$kshp :: nat \Rightarrow hyperproperty \Rightarrow bool$   
 $kshp\ k\ S \triangleq$   
 $S \in HP \wedge$   
 $(\forall T \in Prop. T \notin S \longrightarrow$   
 $(\exists M \in Obs. M \leq T \wedge card\ M = k \wedge$   
 $(\forall T' \in Prop. M \leq T' \longrightarrow T' \notin S)))$   
 $KSHP :: nat \Rightarrow hyperproperty\ set$   
 $KSHP\ k \triangleq \{S. kshp\ k\ S\}$

$fromSome :: 'a\ option \Rightarrow 'a$   
 $fromSome\ x \triangleq (case\ x\ of\ Some\ e \Rightarrow e \mid None \Rightarrow arbitrary)$   
 $fromSomeSt :: state\ option \Rightarrow state$   
 $fromSomeSt\ x \triangleq (case\ x\ of\ Some\ s \Rightarrow s \mid None \Rightarrow \perp)$

$zipn :: nat \Rightarrow (state\ llist)\ llist \Rightarrow (state\ llist)\ llist \Rightarrow bool$   
 $zipn\ k\ T\ t \triangleq$   
 $\forall j :: nat. j < k \longrightarrow t!!j = Some\ (lmap\ (\lambda t. fromSomeSt\ (t!!j))\ T)$   
 — The zip relation. We get unzip for free.

$set-to-llist :: 'a\ set \Rightarrow 'a\ llist$   
 $set-to-llist\ S \triangleq SOME\ l. lset\ l = S$

Following are various axioms about the *zip* operator. Each axiom corresponds to an unproved fact about the operator.

**axioms**

*zip-of-Obs-exists:*

$M \in Obs \Longrightarrow \exists m. zipn\ k\ (set-to-llist\ M)\ m$

— Any observation can be zipped. This axiom is used in the if direction of theorem 3.

*zip-EX-suffix:*

$\llbracket M \in Obs; S \in Systems; zipn\ k\ (set-to-llist\ M)\ m; M \leq S \rrbracket$

$\Longrightarrow \exists s \in kProd\ k\ S. prefix-k\ k\ m\ s$

— There is a suffix  $S^k$  to any zip of an observation, if the system  $S$  is a suffix of the observation.

*zip-of-Obs-fin:*

$\llbracket M \in \text{Obs}; \text{zipn } k \text{ (set-to-llist } M) \text{ } m \rrbracket$

$\implies m \in (\Sigma^*)^*$

— Zipping an observation produces a finite trace over  $\Sigma^k$ .

*unzipped-recoverable:*

$\text{zipn } k \text{ (set-to-llist } M) \text{ } Mz$

$\implies \forall j < k. \exists m \in M. m = \text{lmap } (\lambda t. \text{fromSome } (t!!j)) \text{ } Mz$

— Every member from an unzipped trace set corresponds to some element of the zip.

*unzip-monotonic-wrt-prefix-k:*

$\llbracket \text{zipn } k \text{ (set-to-llist } M) \text{ } Mz; \text{zipn } k \text{ (set-to-llist } T) \text{ } Tz; \text{prefix-k } k \text{ } Mz \text{ } Tz \rrbracket$

$\implies M \leq Tl$

— Unzipping is monotonic.

### constdefs

$\text{noBot} :: \text{state llist} \Rightarrow \text{bool}$

$\text{noBot} \triangleq \text{fnlsts-rec True } (\lambda s r b. b \wedge (s \neq \perp))$

—  $\text{noBot } t$  asserts that the finite trace  $t$  does not contain  $\perp$ .

$\text{bottoms} :: \text{state llist}$  — infinite list of bottoms

$\text{bottoms} \triangleq \text{lconst } \perp$

$\text{prefix-bottom} :: \text{state llist} \Rightarrow \text{state llist} \Rightarrow \text{bool}$  (**infix**  $\leq_{\perp}$  60)

$t \leq_{\perp} u \triangleq \exists tp. \text{noBot } tp \wedge t \leq tp \text{ @@ bottoms} \wedge tp \leq u$

— Effectively removes the bottoms from the first trace, then compares it to the second.

$\text{prefix-k} :: (\text{state llist}) \text{ llist} \Rightarrow \text{nat} \Rightarrow (\text{state llist}) \text{ llist} \Rightarrow \text{bool}$  ( $- \leq_- -$  60)

$t_k \leq_k u_k \triangleq$

$\forall j. j < k \longrightarrow$

$(\text{lmap } (\lambda t. \text{fromSome } (t!!j)) \text{ } t_k) \leq_{\perp} (\text{lmap } (\lambda t. \text{fromSome } (t!!j)) \text{ } u_k)$

— The input traces are over the alphabet  $\Sigma^k$ . We project the  $j$ th position of each element, which creates two traces each with state elements, and compare those with  $\text{prefix-bottom}$ .

$\text{State-K} :: \text{state llist set}$

$\text{State-K} \triangleq \Sigma^*$

$\text{TraceFin-K} :: \text{state llist llist set}$

$\text{TraceFin-K} \triangleq \text{State-K}^*$

$\text{TraceInf-K} :: \text{state llist llist set}$

$\text{TraceInf-K} \triangleq \text{State-K}^{\omega}$

$\text{Prop-K} :: \text{state llist llist set set}$

$Prop\text{-}K \triangleq Pow\ TraceInf\text{-}K$

A generic definition of safety which takes an alphabet as a parameter. For theorem 2 we require reasoning about traces over  $\Sigma$  and  $\Sigma^k$ .

**constdefs**

$spa :: nat \Rightarrow (state\ llist)\ llist\ set \Rightarrow bool$   
 $spa\ k\ P \triangleq P \in Prop\text{-}K$   
 $\wedge (\forall t \in TraceInf\text{-}K. t \notin P \longrightarrow$   
 $(\exists m \in TraceFin\text{-}K. m \leq_k t \wedge$   
 $(\forall t' \in TraceInf\text{-}K. m \leq_k t' \longrightarrow t' \notin P)))$

$SPA :: nat \Rightarrow (state\ llist)\ llist\ set\ set$   
 $SPA\ k \triangleq \{P. spa\ k\ P\}$

$kProd :: nat \Rightarrow state\ llist\ set \Rightarrow (state\ llist)\ llist\ set$   
 $kProd\ k\ S \triangleq \{t \in TraceInf\text{-}K. \exists S' \in Systems.$   
 $S' \subseteq S \wedge card\ S' = k \wedge zipn\ k\ (set\text{-}to\text{-}llist\ S')\ t\}$   
 —  $k$ -product of a system  $S$ .

$pa\text{-}satisfies :: 'a\ llist\ set \Rightarrow 'a\ llist\ set \Rightarrow bool\ ((- \models -) [80,80] 80)$   
 $pa\text{-}satisfies\text{-}def: ts \models p \triangleq ts \subseteq p$   
 — Whether a set of traces over an alphabet  $'a$  satisfies a property.

$KSP :: nat \Rightarrow (state\ llist)\ llist\ set\ set$   
 $KSP\ k \triangleq SPA\ k$

$Bads\text{-}from\text{-}KSaf :: nat \Rightarrow hyperproperty \Rightarrow trace\ set\ set$   
 $Bads\text{-}from\text{-}KSaf\ k\ KK \triangleq$   
 $\{M \in Obs. card\ M \leq k$   
 $\wedge (\exists T \in Prop. T \notin KK \wedge M \leq T)$   
 $\wedge (\forall T' \in Prop. M \leq T' \longrightarrow T' \notin KK)\}$   
 — Boldface  $M$  in the proof of theorem 2.

$Saf\text{-}from\text{-}KSaf :: nat \Rightarrow hyperproperty \Rightarrow (state\ llist)\ llist\ set$   
 $Saf\text{-}from\text{-}KSaf\ k\ KK \triangleq$   
 $\{t \in TraceInf\text{-}K.$   
 $\neg(\exists M \in Obs. \exists tz \in TraceFin\text{-}K.$   
 $M \in Bads\text{-}from\text{-}KSaf\ k\ KK \wedge zipn\ k\ (set\text{-}to\text{-}llist\ M)\ tz \wedge tz \leq_k t)\}$   
 — Boldface  $K$  in the proof of theorem 2.

**lemma** *Saf-from-KSaf-is-safety:*

**fixes**  $k :: nat$

**assumes**  $KK\text{-}KSHP: KK \in KSHP\ k$

**shows**  $Saf\text{-}from\text{-}KSaf\ k\ KK \in KSP\ k$

**proof** —

```

let ?K = Saf-from-KSaf k KK
have Saf-from-KSaf-st: ?K ∈ Prop-K
  unfolding Saf-from-KSaf-def TraceInf-K-def State-K-def Prop-K-def
  by blast
{
  fix t assume t-st: t ∈ TraceInf-K t ∉ ?K
  then have ∃ M ∈ Obs. M ∈ Bads-from-KSaf k KK
    ∧ (∃ tz ∈ TraceFin-K. zipn k (set-to-llist M) tz ∧ tz ≤k t)
    unfolding Saf-from-KSaf-def TraceInf-K-def TraceFin-K-def State-K-def
    by blast
  then obtain M tz where M-tz-st:
    M ∈ Obs
    M ∈ Bads-from-KSaf k KK
    tz ∈ TraceFin-K
    zipn k (set-to-llist M) tz
    tz ≤k t
    by blast
  {
    fix u assume u-st: u ∈ TraceInf-K tz ≤k u
    hence u ∉ ?K using M-tz-st
      unfolding TraceInf-K-def Saf-from-KSaf-def State-K-def by blast
    }
  hence ∃ tz ∈ TraceFin-K.
    tz ≤k t ∧ (∀ u ∈ TraceInf-K. tz ≤k u ⟶ u ∉ Saf-from-KSaf k KK)
    using M-tz-st unfolding TraceFin-K-def TraceInf-K-def State-K-def
    by blast
  }
}
thus ?K ∈ KSP k
  unfolding KSP-def SPA-def spa-def using Saf-from-KSaf-st by blast
qed

```

**lemma** *trace-set-prefix-transitive*:  
 assumes  $X\text{-}p\text{-}Y: X \leq Y$  and  $Y\text{-}p\text{-}Z: Y \leq Z$   
 shows  $X \leq Z$

**proof**–

```

{
  fix x assume x ∈ X
  then obtain y where y ∈ Y x ≤ y
    using X-p-Y unfolding trace-set-prefix-def by blast
  then obtain z where z ∈ Z y ≤ z
    using Y-p-Z unfolding trace-set-prefix-def by blast
  have x ≤ z using ⟨x ≤ y⟩ ⟨y ≤ z⟩
    by (rule llist-le-trans [of x y z])
  hence ∃ z ∈ Z. x ≤ z using ⟨z ∈ Z⟩ by blast
}

```

```

}
thus  $X \leq Z$  unfolding trace-set-prefix-def by blast
qed

```

## 8.2 Theorem

**theorem** *theorem-2-onlyif*:

**fixes**  $k :: \text{nat}$

**assumes**  $S\text{-Sys}: S \in \text{Systems}$  **and**  $KK\text{-KSHP}: KK \in \text{KSHP } k$

**shows**  $\exists K \in \text{KSP } k. ((S \models (KK :: \text{hyperproperty})) \longrightarrow ((k\text{Prod } k S) \models K))$

**proof**–

**let**  $?K = \text{Saf-from-KSaf } k KK$

**let**  $?MM = \text{Bads-from-KSaf } k KK$

**let**  $?S-k = k\text{Prod } k S$

**have**  $K\text{-is-safety}: ?K \in \text{KSP } k$  **using**  $KK\text{-KSHP}$  **by** (*simp add: Saf-from-KSaf-is-safety*)

**have**  $(S \models (KK :: \text{hyperproperty})) \longrightarrow ((?S-k) \models ?K)$

**proof** (*rule ccontr*)

{

**assume**  $\text{neg}: \neg (S \models (KK :: \text{hyperproperty})) \longrightarrow ((?S-k) \models ?K)$

**hence**  $S\text{-Sat-KK}: S \models KK$  **by** *blast*

**have**  $S-k\text{-Unsat}: \neg ((?S-k) \models ?K)$  **using**  $\text{neg}$  **by** *blast*

**have**  $S\text{-in-KK}: S \in KK$

**using**  $S\text{-Sat-KK}$  **unfolding** *hyperproperty-satisfies-def* .

**have**  $S\text{-unsub-K}: \neg ?S-k \subseteq ?K$  **using**  $S-k\text{-Unsat}$  **unfolding** *pa-satisfies-def* .

**then obtain**  $t$  **where**  $t\text{-st}: t \in ?S-k$   $t \notin ?K$  **by** *blast*

**hence**  $t \in \text{TraceInf-K}$  **unfolding** *kProd-def* **by** *blast*

**then obtain**  $M$   $\text{zip-M}$  **where**  $M\text{-zip-M-st}: M \in \text{Obs}$

$M \in ?MM$

$\text{zipn } k (\text{set-to-llist } M) \text{ zip-M}$

$\text{zip-M} \leq_k t$

**using**  $t\text{-st}$  **unfolding** *Saf-from-KSaf-def* **by** *blast*

**obtain**  $T$  **where**  $T\text{-st}: \text{zipn } k (\text{set-to-llist } T) t$

$T \in \text{Prop}$

$T \subseteq S$

**using**  $\langle t \in ?S-k \rangle$  **unfolding** *kProd-def* *Systems-def* *Prop-def* **by** *blast*

**have**  $M\text{-pfx-T}: M \leq T$  **using**  $\langle \text{zipn } k (\text{set-to-llist } T) t \rangle$

$\langle \text{zipn } k (\text{set-to-llist } M) \text{ zip-M} \rangle$

$\langle \text{zip-M} \leq_k t \rangle$

**by** (*simp add: unzip-monotonic-wrt-prefix-k*)

**hence**  $T \notin KK$  **using**  $\langle M \in ?MM \rangle$   $\langle T \in \text{Prop} \rangle$

**unfolding** *Bads-from-KSaf-def* **by** *blast*

**have**  $T \leq S$  **using**  $T\text{-st}$   $S\text{-Sys}$   $\langle t \in ?S-k \rangle$

**unfolding** *trace-set-prefix-def* *Systems-def* *kProd-def* *zipn-def*

**by** *blast*

**with**  $M\text{-pfx-}T$  **have**  $M\text{-pfx-}S: M \leq S$   
**by** (*rule trace-set-prefix-transitive [of M T S]*)  
  
**have**  $S \in Prop$  **using**  $S\text{-Sys}$  **unfolding**  $Systems\text{-def Prop-def}$  **by** *blast*  
**have**  $S \notin KK$  **using**  $M\text{-zip-}M\text{-st } M\text{-pfx-}S \langle S \in Prop \rangle$  **unfolding**  $Bads\text{-from-}KSaf\text{-def}$   
**by** *blast*  
**with**  $S\text{-in-}KK$  **have** *False* **by** *simp*  
**}**  
**thus**  $\neg (S \models KK \longrightarrow kProd\ k\ S \models ?K) \implies False$  **by** *assumption*  
**qed**  
**thus**  $\exists K \in KSP\ k. S \models KK \longrightarrow kProd\ k\ S \models K$   
**using**  $K\text{-is-safety}$  **by** *blast*  
**qed**

**theorem** *theorem-2-if:*

**fixes**  $k :: nat$   
**assumes**  $S\text{-Sys}: S \in Systems$  **and**  $KK\text{-KSHP}: KK \in KSHP\ k$   
**shows**  $\exists K \in KSP\ k. (((kProd\ k\ S) \models K) \longrightarrow (S \models (KK :: hyperproperty)))$   
**proof** –  
**let**  $?K = Saf\text{-from-}KSaf\ k\ KK$   
**let**  $?M = Bads\text{-from-}KSaf\ k\ KK$   
**let**  $?S\text{-}k = kProd\ k\ S$   
**have**  $K\text{-is-safety}: ?K \in KSP\ k$  **using**  $KK\text{-KSHP}$  **by** (*simp add: Saf-from-KSaf-is-safety*)  
**have**  $((?S\text{-}k \models ?K) \longrightarrow (S \models (KK :: hyperproperty)))$   
**proof** (*rule ccontr*)  
**{** **assume**  $neg: \neg (((?S\text{-}k) \models ?K) \longrightarrow (S \models (KK :: hyperproperty)))$   
**hence**  $?S\text{-}k \subseteq ?K$  **unfolding**  $pa\text{-satisfies-def}$  **by** *simp*  
**have**  $\neg (S \models KK)$  **using**  $neg$  **by** *simp*  
**have**  $S \in Prop$  **using**  $S\text{-Sys}$  **unfolding**  $Prop\text{-def Systems-def}$  **by** *blast*  
**hence**  $S \notin KK$  **using**  $\langle \neg (S \models KK) \rangle$   
**unfolding**  $hyperproperty\text{-satisfies-def}$  **by** *simp*  
  
**hence**  
 $\exists M \in Obs. M \leq S \wedge card\ M = k \wedge$   
 $(\forall T' \in Prop. M \leq T' \longrightarrow T' \notin KK)$   
**using**  $\langle S \in Prop \rangle KK\text{-KSHP}$  **unfolding**  $KSHP\text{-def kshp-def}$   
**by** *blast*  
**then obtain**  $M$  **where**  $M\text{-st}: M \leq S\ card\ M = k\ M \in Obs$   
 $\forall T' \in Prop. M \leq T' \longrightarrow T' \notin KK$  **by** *blast*  
**have**  $\exists m. zipn\ k\ (set\text{-to-}l\text{-list}\ M)\ m$   
**using**  $\langle M \in Obs \rangle$  **by** (*simp add: zip-of-Obs-exists [of M k]*)  
**then obtain**  $m$  **where**  $m\text{-st}: zipn\ k\ (set\text{-to-}l\text{-list}\ M)\ m$  **by** *blast*  
**obtain**  $s$  **where**  $s \in ?S\text{-}k\ m \leq_k s$   
**using**  $\langle M \in Obs \rangle \langle S \in Systems \rangle m\text{-st} \langle M \leq S \rangle$   
**using**  $zip\text{-EX-suffix}$  **by** *best*

```

have  $M \in ?M$  unfolding Bads-from-KSaf-def using  $M\text{-st}$   $\langle S \in Prop \rangle$ 
  by blast
have  $m \in TraceFin\text{-}K$  unfolding TraceFin-K-def
  using  $m\text{-st}$   $\langle M \in Obs \rangle$  zip-of-Obs-fin
  unfolding zipn-def State-K-def Obs-def psi-fin-def
  by blast
have  $s \notin ?K$  unfolding Saf-from-KSaf-def
  using  $\langle M \in Obs \rangle$   $\langle m \in TraceFin\text{-}K \rangle$   $\langle M \in ?M \rangle$   $\langle zipn\ k\ (set\text{-to}\text{-}l\text{-}list\ M)\ m \rangle$ 
   $\langle m \leq_k s \rangle$  by blast
hence  $\neg ?S\text{-}k \subseteq ?K$  using  $\langle s \in ?S\text{-}k \rangle$  by blast
hence False using  $\langle ?S\text{-}k \subseteq ?K \rangle$  by blast
}
thus  $\neg (kProd\ k\ S \models Saf\text{-}from\text{-}KSaf\ k\ KK \longrightarrow S \models KK) \implies False$  by
assumption
qed
thus  $\exists K \in KSP\ k.\ kProd\ k\ S \models K \longrightarrow S \models KK$ 
using K-is-safety by blast
qed

end

```

## References

- [1] Denis L. Bueno and Michael R. Clarkson. Hyperproperties: Verification of proofs. Cornell University Computing and Information Science Technical Report, <http://hdl.handle.net/1813/11153>, July 2008.
- [2] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. Cornell University Computing and Information Science Technical Report, <http://hdl.handle.net/1813/9480>, January 2008.
- [3] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. In *Proc. IEEE Computer Security Foundations Symposium*, pages 51–65, June 2008.
- [4] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer-Verlag, London, Cambridge, Massachusetts, 2002.