Document-preparation

karl

22nd January 2024

Contents

Contents

Introduction

In this thesis, I show the corrspondence between various equivalences popular in the reactive systems community and and coordinates of a price function, as introduced by Benjamin Bisping (citation). I formalised the concepts and proofs discussed in this thesis in the interactive proof assistant Isabelle (citation).

- Reactive Systmes
- what are they
- modelling (via lts etc)
- Semantics of resysts
- Verification
- different notions of equivalence (because of nondeteminism?) -> van glabbeek
- Different definitions of semantics -> HML/relational/...

–> linear-time–branching-time spectrum understood through properties of HML

–> capture expressiveness capabilities of HML formulas via a function

–> Contribution o Paper: The in (citation) introduced expressiveness function and its coordinates captures the linear time branching time spectrum..

- Isabelle:

- formalization of concepts, proofs
- what is isabelle
- difference between mathematical concepts and their implementation?

Foundations

In this chapter, relevant concepts will be introduced as well as formalised in Isabelle.

- mention sources (Ben / Max Pohlmann?)

1 Labelled Transition Systems

A LTS ...

- examples (to reuse later?)??? - Definitions (wøisabelle)?

Isabelle

Zustände: $'s$ und Aktionen $'a$, Transitionsrelation ist locale trans. Ein LTS wird dann durch seine Transitionsrelation definiert.

locale *lts* =
  fixes *tran* :: ‹$'s \Rightarrow 'a \Rightarrow 's \Rightarrow bool$›
    (- ↦- - [*70*, *70*, *70*] *80*)
begin

abbreviation *derivatives* :: ‹$'s \Rightarrow 'a \Rightarrow 's\ set$›
  where
‹$derivatives\ p\ \alpha \equiv \{p'.\ p \mapsto\alpha\ p'\}$›

Transition System is image-finite

definition *image-finite* **where**
‹$image\text{-}finite \equiv (\forall p\ \alpha.\ finite\ (derivatives\ p\ \alpha))$›

definition *image-countable* :: ‹*bool*›
  where ‹$image\text{-}countable \equiv (\forall\ p\ \alpha.\ countable\ (derivatives\ p\ \alpha))$›

stimmt definition? definition benötigt nach umstieg auf sets?

definition *lts-finite* **where**
‹$lts\text{-}finite \equiv (finite\ (UNIV :: 's\ set))$›

abbreviation *initial-actions*:: ‹$'s \Rightarrow 'a\ set$›
  where
‹$initial\text{-}actions\ p \equiv \{\alpha|\alpha.\ (\exists p'.\ p \mapsto\alpha\ p')\}$›

abbreviation *deadlock* :: ‹$'s \Rightarrow bool$› **where**
‹$deadlock\ p \equiv (\forall\, a.\ derivatives\ p\ a = \{\})$›

nötig?

abbreviation *relevant-actions* :: ‹$'a\ set$›
  where
‹$relevant\text{-}actions \equiv \{a.\ \exists\, p\ p'.\ p \mapsto a\ p'\}$›

inductive *step-sequence* :: ‹$'s \Rightarrow\ 'a\ list \Rightarrow\ 's \Rightarrow bool$› (‹- ↦\$ - -›[70,70,70] 80)
where
‹$p \mapsto\$\ []\ p$› |
‹$p \mapsto\$\ (a\#rt)\ p''$› **if** ‹$\exists\, p'.\ p \mapsto\ a\ p' \wedge p' \mapsto\$\ rt\ p''$›

Introduce these definitions later?

abbreviation *traces* :: ‹$'s \Rightarrow\ 'a\ list\ set$› **where**
‹$traces\ p \equiv \{tr.\ \exists\, p'.\ p \mapsto\$\ tr\ p'\}$›

abbreviation *all-traces* :: $'a\ list\ set$ **where**
$all\text{-}traces \equiv\{tr.\ \exists\, p\ p'.\ p \mapsto\$\ tr\ p'\}$

inductive *paths*:: ‹$'s \Rightarrow\ 's\ list \Rightarrow\ 's \Rightarrow bool$› **where**
‹$paths\ p\ []\ p$› |
‹$paths\ p\ (a\#as)\ p''$› **if** $\exists\, \alpha.\ p \mapsto\ \alpha\ a \wedge (paths\ a\ as\ p'')$

lemma *path-implies-seq*:
  assumes *A1*: $\exists\, xs.\ paths\ p\ xs\ p'$
  shows $\exists\, ys.\ p \mapsto\$\ ys\ p'$
⟨*proof*⟩

lemma *seq-implies-path*:
  assumes *A1*: $\exists\, ys.\ p \mapsto\$\ ys\ p'$
  shows $\exists\, xs.\ paths\ p\ xs\ p'$
⟨*proof*⟩

Trace preorder as inclusion of trace sets

definition *trace-preordered* (**infix** ‹$\lesssim T$› 60)**where**
‹$trace\text{-}preordered\ p\ q \equiv traces\ p \subseteq traces\ q$›

Trace equivalence as mutual preorder

abbreviation *trace-equivalent* (**infix** ‹$\simeq T$› 60) **where**
‹$p \simeq T\ q \equiv p \lesssim T\ q \wedge q \lesssim T\ p$›

Trace preorder is transitive

lemma *T-trans*:
  shows ‹$transp\ (\lesssim T)$›
  ⟨*proof*⟩

Failure Pairs

abbreviation *failure-pairs* :: ‹$'s \Rightarrow ('a\ list \times 'a\ set)\ set$›
where
‹$failure\text{-}pairs\ p \equiv \{(xs, F)|xs\ F.\ \exists p'.\ p \mapsto\$\ xs\ p' \wedge (initial\text{-}actions\ p' \cap F = \{\})\}$›

Failure preorder and -equivalence

definition *failure-preordered* (**infix** ‹$\lesssim F$› *60*) **where**
‹$p \lesssim F\ q \equiv failure\text{-}pairs\ p \subseteq failure\text{-}pairs\ q$›

abbreviation *failure-equivalent* (**infix** ‹$\simeq F$› *60*) **where**
‹ $p \simeq F\ q \equiv p \lesssim F\ q \wedge q \lesssim F\ p$›

Possible future sets

abbreviation *possible-future-pairs* :: ‹$'s \Rightarrow ('a\ list \times 'a\ list\ list)\ set$›
where
‹$possible\text{-}future\text{-}pairs\ p \equiv \{(xs, X)|xs\ X.\ \exists p'.\ p \mapsto\$\ xs\ p' \wedge traces\ p' = (set\ X)\}$›

definition *possible-futures-equivalent* (**infix** ‹$\simeq PF$› *60*) **where**
‹$p \simeq PF\ q \equiv (possible\text{-}future\text{-}pairs\ p = possible\text{-}future\text{-}pairs\ q)$›

lemma *PF-trans*: *transp* ($\simeq PF$)
⟨*proof*⟩

isomorphism

definition *isomorphism* :: ‹$('s \Rightarrow 's) \Rightarrow bool$› **where**
‹$isomorphism\ f \equiv bij\ f \wedge (\forall p\ a\ p'.\ p \mapsto a\ p' \longleftrightarrow f\ p \mapsto a\ (f\ p'))$›

definition *is-isomorphic* :: ‹$'s \Rightarrow 's \Rightarrow bool$› (**infix** ‹$\simeq ISO$› *60*) **where**
‹$p \simeq ISO\ q \equiv \exists f.\ isomorphism\ f \wedge (f\ p) = q$›

Two states are simulation preordered if they can be related by a simulation relation. (Implied by isometry.)

definition *simulation*
where ‹$simulation\ R \equiv$
$\forall p\ q\ a\ p'.\ p \mapsto a\ p' \wedge R\ p\ q \longrightarrow (\exists q'.\ q \mapsto a\ q' \wedge R\ p'\ q')$›

definition *simulated-by* (**infix** ‹$\lesssim S$› *60*)
where ‹$p \lesssim S\ q \equiv \exists R.\ R\ p\ q \wedge simulation\ R$›

Two states are bisimilar if they can be related by a symmetric simulation.

definition *bisimilar* (**infix** ‹$\simeq B$› *80*) **where**
‹$p \simeq B\ q \equiv \exists R.\ simulation\ R \wedge symp\ R \wedge R\ p\ q$›

Bisimilarity is a simulation.

lemma *bisim-sim*:
shows ‹*simulation* ($\simeq B$)›
⟨*proof*⟩

end
end