COMP3610/6361
Principles of Programming Languages

Peter Höfner

Oct 20, 2023

Section 1

Semantic Equivalences

Labelled Transition Systems

CCS naturally implies a graphical model of computation.

a **labelled transition system** (LTS) is a pair $(S, \Rightarrow)$ with

- S a set (of *states* or processes), and
- $\Rightarrow \subseteq S \times Act \times S$, the *transition relation*.

here $Act = A \uplus \{\tau\}$ is a set of *actions*, containing visible actions $a, b, c, ... \in A$, and the *invisible action* τ.

a finite *path* is a sequence $p_0 \xrightarrow{\lambda_1} p_1 \xrightarrow{\lambda_2} \cdots \xrightarrow{\lambda_n} p_n$ with $p_i \in S$ for $i = 0, ..., n$ and $(p_{i-1}, \lambda_i, p_i) \in \Rightarrow$ for all $i = 1, ..., n$.

Trace equivalence

- if such a path exists, then the sequence $\lambda_1 \lambda_2 \ldots \lambda_n$ is a (partial) *trace* of the process p_0
- two processes p and q are (partial) *trace equivalent* if they have the same (partial) traces.

Four Kinds of Trace Equivalence

Let $T^*(p)$ be the set of (partial) traces of process $p \in S$.
Let $T^\infty(p)$ be the set of infinite traces of p.
Let $CT^*(p)$ be the set of completed traces of p.
Let $CT^\infty(p) := CT^*(p) \uplus T^\infty(p)$.

A finite trace is *complete* if it last state has no outgoing transition.

Write $p =^*_T q$ if $T^*(p) = T^*(q)$ — (partial) trace equivalence.
Write $p =^*_{CT} q$ if $CT^*(p) = CT^*(q)$ and $T^*(p) = T^*(q)$ — completed trace equivalence
Write $p =^\infty_T q$ if $T^\infty(p) = T^\infty(q)$ and $T^*(p) = T^*(q)$ — infinitary trace equivalence
Write $p =^\infty_{CT}$ if $CT^\infty(p) = CT^\infty(q)$ — infinitary completed tr. eq.

A Lattice of Semantic Equivalence Relations

A relation $\sim \subseteq S \times S$ on processes is an *equivalence relation* if it is

- *reflexive*: $p \sim p$,
- *symmetric*: if $p \sim q$ then $q \sim p$,
- and *transitive*: if $p \sim q$ and $q \sim r$ then $p \sim r$.

Let $[p]_\sim$ be the *equivalence class* of p: the set of all processes that are $\sim$-equivalent to p.

$$[p]_\sim := \{q \in S \mid q \sim p\}.$$

Equivalence relation $\sim$ is *finer than* equivalence relation $\approx$ iff

$$p \sim q \Rightarrow p \approx q.$$

Thus if $\sim \subseteq \approx$. In that case each equivalence class of $\sim$ is included in an equivalence class of $\approx$.

Four Additional Trace Equivalence

A *weak* trace is obtained from a strong one by deleting all τs.
Let $WT^*(p) := \{\textit{detau}(\sigma) \mid \sigma \in T^*(p)\}$.

This leads to *weak trace equivalences* $=^*_{WT}$, $=^\infty_{WT}$, $=^*_{WCT}$, $=^\infty_{WCT}$.

Safety and Liveness Properties

A **safety** property says that something bad will never happen.
A **liveness** property says that something good will happen eventually.

If we deem two processes p and q semantically equivalent we often want them to have the same safety and/or liveness properties.

$$ab \stackrel{?}{\sim} ab + a$$

Weak partial trace equivalence respects safety properties.

$$ag \stackrel{?}{\sim} ag + a$$

We need at least completed traces to deal with liveness properties

Compositionality

If $p \sim q$ then $C[p] \sim C[q]$.
Here $C[\ ]$ is a context, made from operators of some language.

For instance $(\_|\bar{b}.\bar{a}.\mathbf{nil})\backslash\{a,b\}$ is a CCS-context.
If $p \sim q$ then $(p|\bar{b}.\bar{a}.\mathbf{nil})\backslash\{a,b\} \sim (q|\bar{b}.\bar{a}.\mathbf{nil})\backslash\{a,b\}$.

Then $\sim$ is a *congruence* for the language,
or the language if *compositional* for $\sim$.

$$p \sim p' \quad \Rightarrow \quad (p|p|...|p)\backslash L \sim (p'|p'|...|p')\backslash L.$$

$a.b + a.c =^*_{CT} a.(b + c)$ but

$((a.b + a.c)|\bar{a}.\bar{b})\backslash\{a,b\} \neq^*_{CT} (a.(b + c)|\bar{a}.\bar{b})\backslash\{a,b\}$.

Thus $=^*_{CT}$ is a not a congruence for CCS.

Congruence closure

Theorem: Given any equivalence $\approx$ that need not be a congruence for some language $\mathcal{L}$, there exists a coarsest congruence $\sim$ for $\mathcal{L}$ that is finer than $\sim$.

In fact, $\sim$ can be defined by

$$p \sim q \quad :\Leftrightarrow \quad C[p] \approx C[q] \text{ for any } \mathcal{L}\text{-context } C[\ ].$$

Bisimulation equivalence

A relation $\mathcal{R} \subseteq S \times S$ is a *bisimulation* if it satisfies:

- if $p\mathcal{R}q$ and $p \xrightarrow{\lambda} p'$ then $\exists q'$ s.t. $q \xrightarrow{\lambda} q'$ and $p'\mathcal{R}q'$, and
- if $p\mathcal{R}q$ and $q \xrightarrow{\lambda} q'$ then $\exists p'$ s.t. $p \xrightarrow{\lambda} p'$ and $p'\mathcal{R}q'$.

Two processes $p, q \in S$ are *bisimulation equivalent* or *bisimilar* —notation $p =_B q$—if $p\mathcal{R}q$ for some bisimulation $\mathcal{R}$.

Examples: $a.b + a.c \neq_B a.(b + c)$ $a.b + a.b =_B a.b$

Weak bisimulation equivalence

A relation $\mathcal{R} \subseteq S \times S$ is a *weak bisimulation* if it satisfies:

- if $p\mathcal{R}q$ and $p \xrightarrow{\lambda} p'$ then $\exists q'$ s.t. $q \Longrightarrow\!\xrightarrow{(\lambda)}\!\Longrightarrow q'$ and $p'\mathcal{R}q'$, and
- if $p\mathcal{R}q$ and $q \xrightarrow{\lambda} q'$ then $\exists p'$ s.t. $p \Longrightarrow\!\xrightarrow{(\lambda)}\!\Longrightarrow p'$ and $p'\mathcal{R}q'$.

Here $\Longrightarrow$ denotes a finite sequence of τ-steps,
and (λ) means λ, except that it is optional in case $\lambda = \tau$.
(That is, $p \xrightarrow{(\lambda)} q$ iff $p \xrightarrow{\lambda} q \vee (\lambda = \tau \wedge q = p)$.)
Two processes $p, q \in S$ are *weakly bisimilar*
—notation $p =_{WB} q$—if $p\mathcal{R}q$ for some bisimulation $\mathcal{R}$.

Examples: $\tau.b + c \neq_{WB} b + c$ $\tau.b + b =_{WB} b$

Semantic Equivalences – Summary

- relate to systems (via LTSs)
- can be extended to states carrying stores
- sos-rules give raise to LTSs in a straightforward way
- reduce complicated (big) systems to simpler ones
- smaller systems may be easier to verify
- understand which properties are preserved