PSC 2023/24 (375AA, 9CFU)

Principles for Software Composition

Roberto Bruni

http://www.di.unipi.it/~bruni/

http://didawiki.di.unipi.it/doku.php/magistraleinformatica/psc/start

19 - Hennessy-Milner Logic
CCS syntax

\[ p, q ::= \begin{array}{ll}
\text{nil} & \text{inactive process} \\
\text{x} & \text{process variable (for recursion)} \\
\mu.p & \text{action prefix} \\
\text{p \setminus x} & \text{restricted channel} \\
\text{p[\phi]} & \text{channel relabelling} \\
\text{p + q} & \text{nondeterministic choice (sum)} \\
\text{p|q} & \text{parallel composition} \\
\text{rec} x. \ p & \text{recursion}
\end{array} \]

(operators are listed in order of precedence)
CCS op. semantics

Act) $\mu.p \xrightarrow{\mu} p$

Res) $p \xrightarrow{\mu} q$ $\mu \notin \{\alpha, \overline{\alpha}\}$

$p \xrightarrow{\mu} q \quad p \setminus \alpha \xrightarrow{\mu} q \setminus \alpha$

Rel) $p \xrightarrow{\mu} q$

$p[\phi] \xrightarrow{\phi(\mu)} q[\phi]$

SumL) $p_1 \xrightarrow{\mu} q$

$p_1 + p_2 \xrightarrow{\mu} q$

SumR) $p_2 \xrightarrow{\mu} q$

$p_1 + p_2 \xrightarrow{\mu} q$

ParL) $p_1 \xrightarrow{\mu} q_1$

$p_1 | p_2 \xrightarrow{\mu} q_1 | p_2$

Com) $p_1 \xrightarrow{\lambda} q_1$

$p_2 \xrightarrow{\overline{\lambda}} q_2$

$p_1 | p_2 \xrightarrow{\tau} q_1 | q_2$

ParR) $p_2 \xrightarrow{\mu} q_2$

$p_1 | p_2 \xrightarrow{\mu} p_1 | q_2$

Rec) $p[\text{rec } x. \ p/x] \xrightarrow{\mu} q$

$\text{rec } x. \ p \xrightarrow{\mu} q$
HML
Hennessy-Milner Logic
From your forms

Modal and temporal logics

(over 8 answers)
Logical equivalence

Let us take another approach to equivalence

we define some logic (set of formulas)

- a process may or may not satisfy a formula
- two processes are (logically) equivalent when they satisfy exactly the same formulas

formulas must describe behavioural properties of processes
the ability / inability to perform transitions (modal logic: possibly, necessarily)

then, we can compose formulas with usual operators
Hennessy–Milner Logic

We present the core operators

multi-modal:
modal operators are parameterised by actions

no negation:
the converse of a formula can also be written as a formula

no recursion:
each formula express properties about finite steps ahead

denotational semantics of a formula (postponed):
set of processes that satisfy the formula
HML: syntax

\[ F, G \quad ::= \quad \text{tt} \quad \text{true} \]
\[ \quad | \quad \text{ff} \quad \text{false} \]
\[ \quad | \quad \bigwedge_{i \in I} F_i \quad \text{conjunction} \]
\[ \quad | \quad \bigvee_{i \in I} F_i \quad \text{disjunction} \]
\[ \quad | \quad \diamond_{\mu} F \quad \text{diamond operator} \quad \langle \mu \rangle F \]
\[ \quad | \quad \Box_{\mu} F \quad \text{box operator} \quad [\mu] F \]

\[ \mathcal{L} \quad \text{set of all formulas} \]
HML: semantics

\[ p \models F \] reads “\( p \) satisfies \( F \)”

defined inductively on the structure of the formula

\[ p \models \texttt{tt} \] any process satisfies true
(no process satisfies false)

\[ p \models \bigwedge_{i \in I} F_i \] iff \( \forall i \in I. \ p \models F_i \) \quad \text{\( p \) satisfies all \( F_i \)}

\[ p \models \bigvee_{i \in I} F_i \] iff \( \exists i \in I. \ p \models F_i \) \quad \text{\( p \) satisfies one of the \( F_i \)}

\[ p \models \Diamond_{\mu} F \] iff \( \exists p'. \ p \xrightarrow{\mu} p' \land p' \models F \) \quad \text{\( p \) can make one \( \mu \)-step and then satisfy \( F \)}

\[ p \models \Box_{\mu} F \] iff \( \forall p'. \ p \xrightarrow{\mu} p' \Rightarrow p' \models F \) \quad \text{\( F \) is satisfied after any \( \mu \)-step of \( p \)}
Examples

\(\Diamond_\alpha \triangledown\) satisfied by any process that can make an \(\alpha\)-step

\(\Box_\beta \lozenge\) satisfied by any process that cannot make a \(\beta\)-step

\(\Diamond_\alpha \lozenge\) same as \(\lozenge\)
  if a process cannot do \(\alpha\) the modality is missed
  if a process can do \(\alpha\) its continuation cannot satisfy \(\lozenge\)

\(\Box_\beta \triangledown\) same as \(\triangledown\)
  if a process cannot do \(\beta\) the modality holds trivially
  if a process does \(\beta\) its continuation will satisfy \(\triangledown\)

\(\Diamond_\alpha (\Diamond_\beta \triangledown \land \Box_\gamma \lozenge)\) satisfied by any process the can do \(\alpha\)
  and reach a process that can do \(\beta\) but not \(\gamma\)
Examples

\[ \begin{align*}
    p & \models \diamondsuit_\alpha \text{tt} & \checkmark \\
    p & \models \square_\alpha \diamondsuit_\beta \text{tt} & \times \\
    p & \models \diamondsuit_\alpha \square_\beta \text{ff} \land \diamondsuit_\alpha \square_\gamma \text{ff} & \checkmark \\
    p & \models \square_\alpha (\diamondsuit_\beta \text{tt} \lor \diamondsuit_\gamma \text{tt}) & \checkmark \\
    p & \models \square_\alpha (\diamondsuit_\beta \text{tt} \land \diamondsuit_\gamma \text{tt}) & \times \\
    p & \models \diamondsuit_\alpha (\diamondsuit_\beta \text{tt} \land \diamondsuit_\gamma \text{tt}) & \checkmark
\end{align*} \]
Box/diamond duality

\[ \neg \Box_{\mu} F \]

\[ \equiv \neg (\exists p'. \ p \overset{\mu}{\rightarrow} p' \land p' \models F) \]

\[ \equiv \forall p'. \ \neg (p \overset{\mu}{\rightarrow} p' \land p' \models F) \]

\[ \equiv \forall p'. \ \neg (p \overset{\mu}{\rightarrow} p') \lor \neg (p' \models F) \]

\[ \equiv \forall p'. \ p \overset{\mu}{\rightarrow} p' \Rightarrow (p' \models \neg F) \]

\[ \equiv \Box_{\mu} \neg F \]
Negation

not present in the syntax, but not needed

any formula $F$ has a converse formula $F^c$ such that

$$\forall p. \ p \models F \iff p \not\models F^c$$

$F^c$ can be defined by structural induction

$$tt^c \triangleq ff$$

$$(\bigwedge_{i \in I} F_i)^c \triangleq \bigvee_{i \in I} F_i^c$$

$$(\Box_\mu F)^c \triangleq \Box_\mu F^c$$

$$ff^c \triangleq tt$$

$$(\bigvee_{i \in I} F_i)^c \triangleq \bigwedge_{i \in I} F_i^c$$

$$(\Diamond_\mu F)^c \triangleq \Diamond_\mu F^c$$

example

$$(\Diamond_\alpha tt)^c = \Box_\alpha tt^c = \Box_\alpha ff$$

(can do $\alpha^c$ = cannot do $\alpha$)
Extended syntax

\[ A = \{ \mu_1, \ldots, \mu_n \} \]

\[
\Diamond_A F \triangleq \bigvee_{i \in [1,n]} \Diamond_{\mu_i} F \\
\Box_A F \triangleq \bigwedge_{i \in [1,n]} \Box_{\mu_i} F
\]

\[
\Diamond_{\emptyset} F \triangleq \mathbf{ff} \\
\Box_{\emptyset} F \triangleq \mathbf{tt}
\]
HML: logical equivalence

two processes are equivalent iff they satisfy the same formulas

\[ p \equiv_{\text{HM}} q \quad \text{iff} \quad \forall F \in \mathcal{L}. \ (p \models F \iff q \models F) \]
Strong bis as logic equiv

**TH.** for any finitely branching processes $p, q$

\[ p \simeq q \iff p \equiv_{HM} q \]

(proof omitted)

consequences:

to show that two processes are strong bisimilar:
exhibit a strong bisimulation relation that relates them

to show that two processes are not strong bisimilar:
exhibit a HML formula that distinguishes between them
Exercise

find a HML formula that distinguishes the two processes

\[
P_0 \xrightarrow{\beta} \text{nil} \quad P_0 \not\equiv R_0
\]

\[
P_1 \xrightarrow{\beta} \text{nil}
\]

\[
P_2 \xrightarrow{\beta} \text{nil}
\]

\[
P_0 \models F
\]

\[
F \triangleq \lozenge_\alpha \lozenge_\alpha \lozenge_\alpha \top \top
\]

\[
R_0 \not\models F
\]
Exercise

find a HML formula that distinguishes the two processes

\[
P_0 \xrightarrow{\beta} \text{nil} \quad P_0 \not\equiv Q_0
\]

\[
\begin{array}{c}
P_0 \\
\beta \\
\alpha \\
\beta \\
\alpha \\
\beta \\
\alpha
\end{array}
\]

\[
P_1
\]

\[
P_2
\]

\[
P_0 \models F
\]

\[
F \triangleq \Diamond \alpha \Box \alpha \Diamond \alpha \top \top
\]

\[
Q_0 \xrightarrow{\beta} \text{nil}
\]

\[
\begin{array}{c}
Q_0 \\
\beta \\
\alpha \\
\beta \\
\alpha \\
\beta \\
\alpha
\end{array}
\]

\[
\begin{array}{c}
Q_1 \\
\beta \\
\alpha
\end{array}
\]

\[
Q_2
\]

\[
Q_0 \not\models F
\]