PSC 2022/23 (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

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

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

\[
\begin{align*}
\text{Act}) & \quad \mu \cdot p \xrightarrow{\mu} p \\
\text{Res}) & \quad p \xrightarrow{\mu} q, \mu \not\in \{\alpha, \overline{\alpha}\} \\
& \quad p \setminus \alpha \xrightarrow{\mu} q \setminus \alpha \\
\text{Rel}) & \quad p \xrightarrow{\mu} q \\
& \quad p[\phi] \xrightarrow{\phi(\mu)} q[\phi] \\
\text{SumL}) & \quad p_1 \xrightarrow{\mu} q \\
& \quad p_1 + p_2 \xrightarrow{\mu} q \\
\text{SumR}) & \quad p_2 \xrightarrow{\mu} q \\
& \quad p_1 + p_2 \xrightarrow{\mu} q \\
\text{ParL}) & \quad p_1 \xrightarrow{\mu} q_1 \\
& \quad p_1 | p_2 \xrightarrow{\mu} q_1 | q_2 \\
\text{Com}) & \quad p_1 \xrightarrow{\lambda} q_1, p_2 \xrightarrow{\overline{\lambda}} q_2 \\
& \quad p_1 | p_2 \xrightarrow{\tau} q_1 | q_2 \\
\text{ParR}) & \quad p_2 \xrightarrow{\mu} q_2 \\
& \quad p_1 | p_2 \xrightarrow{\mu} p_1 | q_2 \\
\text{Rec}) & \quad p[\text{rec } x \cdot p/x] \xrightarrow{\mu} q \\
& \quad \text{rec } x \cdot p \xrightarrow{\mu} q
\end{align*}
\]
HML
Hennessy-Milner Logic
From your forms

(over 15 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
\[ F, G \ ::= \ \begin{array}{ll}
\text{tt} & \text{true} \\
\text{ff} & \text{false} \\
\bigwedge_{i \in I} F_i & \text{conjunction} \\
\bigvee_{i \in I} F_i & \text{disjunction} \\
\Diamond_{\mu} F & \text{diamond operator } \langle \mu \rangle F \\
\Box_{\mu} F & \text{box operator } [\mu] F
\end{array} \]

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

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

defined inductively on the structure of the formula

**HML Semantics**

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

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

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

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

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

$\Diamond_\alpha \text{tt}$ satisfied by any process that can make an $\alpha$-step

$\Box_\beta \text{ff}$ satisfied by any process that cannot make a $\beta$-step

$\Diamond_\alpha \text{ff}$ same as $\text{ff}$
   if a process cannot do $\alpha$ the modality is missed
   if a process can do $\alpha$ its continuation cannot satisfy $\text{ff}$

$\Box_\beta \text{tt}$ same as $\text{tt}$
   if a process cannot do $\beta$ the modality holds trivially
   if a process does $\beta$ its continuation will satisfy $\text{tt}$

$\Diamond_\alpha (\Diamond_\beta \text{tt} \land \Box_\gamma \text{ff})$ satisfied by any process the can do $\alpha$ and reach a process that can do $\beta$ but not $\gamma$
Examples

\[ p \models \Diamond_\alpha \mathtt{tt} \quad \checkmark \]

\[ p \models \Box_\alpha \Diamond_\beta \mathtt{tt} \quad \xmark \]

\[ p \models \Diamond_\alpha \Box_\beta \mathtt{ff} \land \Diamond_\alpha \Box_\gamma \mathtt{ff} \quad \checkmark \]

\[ p \models \Box_\alpha (\Diamond_\beta \mathtt{tt} \lor \Diamond_\gamma \mathtt{tt}) \quad \checkmark \]

\[ p \models \Box_\alpha (\Diamond_\beta \mathtt{tt} \land \Diamond_\gamma \mathtt{tt}) \quad \xmark \]

\[ p \models \Diamond_\alpha (\Diamond_\beta \mathtt{tt} \land \Diamond_\gamma \mathtt{tt}) \quad \checkmark \]
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

\[ \begin{align*}
\tt^c & \triangleq \ff \\
(\bigwedge_{i \in I} F_i)^c & \triangleq \bigvee_{i \in I} F_i^c \\
(\Diamond_\mu F)^c & \triangleq \Box_\mu F^c \\
\ff^c & \triangleq \tt \\
(\bigvee_{i \in I} F_i)^c & \triangleq \bigwedge_{i \in I} F_i^c \\
(\Box_\mu F)^c & \triangleq \Diamond_\mu F^c
\end{align*} \]

example

\[ \Diamond_\alpha \tt = \Box_\alpha \tt^c = \Box_\alpha \ff \]

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

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

\[ \Diamond_A F \triangleq \Diamond \mu_1 F \lor \cdots \lor \Diamond \mu_n F \quad \Box_A F \triangleq \Box \mu_1 F \land \cdots \land \Box \mu_n F \]

\[ = \bigvee_{i \in [1,n]} \Diamond \mu_i F \quad = \bigwedge_{i \in [1,n]} \Box \mu_i F \]

\[ \Diamond \emptyset F \triangleq \mathbf{ff} \quad \Box \emptyset F \triangleq \mathbf{tt} \]
HML: logical equivalence

Two processes are equivalent iff they satisfy the same formulas:

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

**Formulas:**

- \( p \models F \)
- \( q \notmodels F \)
- \( p \notmodels F^c \)
- \( q \models F^c \)

**Formal Definitions:**

- \( F \triangleq \Diamond_{\alpha}(\Diamond_{\beta}tt \land \Diamond_{\gamma}tt) \)
- \( F^c \triangleq \Box_{\alpha}(\Box_{\beta}ff \lor \Box_{\gamma}ff) \)
Strong bis as logic equiv

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

$$p \simeq q \iff p \equiv_{\text{HM}} q$$

(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

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

\[ R_0 \xrightarrow{\beta} \text{nil} \]

\[ P_0 \models F \]

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

\[ R_0 \not\models F \]
Exercise

find a HML formula that distinguishes the two processes

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

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

\[
P_0 \models F \quad F \triangleq \Diamond_\alpha \Box_\alpha \Diamond_\alpha tt \quad Q_0 \not\models F
\]