Semantics and Verification 2006

Lecture 5

- Hennessy-Milner logic
- syntax and semantics
- correspondence with strong bisimilarity
- examples in CWB

Verifying Correctness of Reactive Systems

Let Impl be an implementation of a system (e.g. in CCS syntax).

Equivalence Checking Approach

 $Impl \equiv Spec$

- \equiv is an abstract equivalence, e.g. \sim or \approx
- Spec is often expressed in the same language as Impl
- Spec provides the full specification of the intended behaviour

Model Checking Approach

 $Impl \models Property$

- \models is the satisfaction relation
- Property is a particular feature, often expressed via a logic
- Property is a partial specification of the intended behaviour

Equivalence Checking vs. Model Checking Modal and Temporal Properties

Model Checking of Reactive Systems

Our Aim

Develop a logic in which we can express interesting properties of reactive systems.

Equivalence Checking vs. Model Checking Modal and Temporal Properties

Logical Properties of Reactive Systems

Modal Properties – what can happen **now** (possibility, necessity)

- drink a coffee (can drink a coffee now)
- does not drink tea
- drinks both tea and coffee
- drinks tea after coffee

Temporal Properties – behaviour in time

- never drinks any alcohol (safety property: nothing bad can happen)
- eventually will have a glass of wine (liveness property: something good will happen)

Can these properties be expressed using equivalence checking?

Syntax Semantics Negation in Hennessy-Milner Logic Denotational Semantics

Hennessy-Milner Logic – Syntax

Syntax of the Formulae $(a \in Act)$

 $F, G ::= tt \mid ff \mid F \land G \mid F \lor G \mid \langle a \rangle F \mid [a]F$

Intuition:

- tt all processes satisfy this property
- ff no process satisfies this property
- $\wedge,\,\vee\,$ usual logical AND and OR
- $\langle a \rangle F$ there is at least one *a*-successor that satisfies F
- [a]F all *a*-successors have to satisfy F

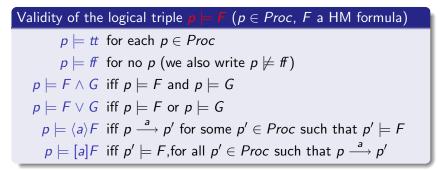
Remark

Temporal properties like *always/never in the future* or *eventually* are not included.

Syntax Semantics Negation in Hennessy-Milner Logic Denotational Semantics

Hennessy-Milner Logic – Semantics

Let $(Proc, Act, \{\stackrel{a}{\longrightarrow} | a \in Act\})$ be an LTS.



We write $p \not\models F$ whenever p does not satisfy F.

Syntax Semantics Negation in Hennessy-Milner Logic Denotational Semantics

What about Negation?

For every formula F we define the formula F^c as follows:

- $tt^c = ff$
- *ff*^c = *tt*
- $(F \wedge G)^c = F^c \vee G^c$
- $(F \vee G)^c = F^c \wedge G^c$
- $(\langle a \rangle F)^c = [a]F^c$

•
$$([a]F)^c = \langle a \rangle F^c$$

Theorem (F^c is equivalent to the negation of F)

For any $p \in Proc$ and any HM formula F

$$p \not\models F \Longrightarrow p \models F^c$$

Hennessy-Milner Logic – Denotational Semantics

For a formula F let $\llbracket F \rrbracket \subseteq Proc$ contain all states that satisfy F.

Denotational Semantics: $\llbracket_{-}\rrbracket$: Formulae $\rightarrow 2^{Proc}$

- $\llbracket F \lor G \rrbracket = \llbracket F \rrbracket \cup \llbracket G \rrbracket$
- $\llbracket F \land G \rrbracket = \llbracket F \rrbracket \cap \llbracket G \rrbracket$
- $\llbracket \langle a \rangle F \rrbracket = \langle \cdot a \cdot \rangle \llbracket F \rrbracket$
- $\llbracket [a]F \rrbracket = [\cdot a \cdot] \llbracket F \rrbracket$

where $\langle \cdot a \cdot \rangle, [\cdot a \cdot] : 2^{(Proc)} \rightarrow 2^{(Proc)}$ are defined by

$$\langle \cdot a \cdot \rangle S = \{ p \in Proc \mid \exists p'. p \xrightarrow{a} p' \text{ and } p' \in S \}$$

 $[\cdot a \cdot]S = \{ p \in Proc \mid \forall p'. p \xrightarrow{a} p' \implies p' \in S \}.$

Syntax Semantics Negation in Hennessy-Milner Logic Denotational Semantics

The Correspondence Theorem

Theorem

Let $(Proc, Act, \{ \xrightarrow{a} | a \in Act \})$ be an LTS, $p \in Proc$ and F a formula of Hennessy-Milner logic. Then

 $p \models F$ if and only if $p \in \llbracket F \rrbracket$.

Proof: by structural induction on the structure of the formula F.

Image-Finite Labelled Transition Systems Hennessy-Milner Theorem Example Sessions in CWB

Image-Finite Labelled Transition System

Image-Finite System

Let $(Proc, Act, \{\stackrel{a}{\longrightarrow} | a \in Act\})$ be an LTS. We call it image-finite iff for every $p \in Proc$ and every $a \in Act$ the set

$$\{p' \in Proc \mid p \stackrel{a}{\longrightarrow} p'\}$$

is finite.

Relationship between HM Logic and Strong Bisimilarity

Theorem (Hennessy-Milner)

Let $(Proc, Act, \{\stackrel{a}{\longrightarrow} | a \in Act\})$ be an image-finite LTS and $p, q \in St$. Then

$p \sim q$

if and only if

for every HM formula $F: (p \models F \iff q \models F)$.

Image-Finite Labelled Transition Systems Hennessy-Milner Theorem Example Sessions in CWB

CWB Session

borg\$ /pack/FS/CWB/cwb

- > input "hm.cwb";
- > print;
- > help logic;
- > checkprop(S,<a>(T & <c>T));
 true
- > checkprop(T,<a>(T & <c>T));
 false
- > help dfstrong;
- > dfstrong(S,T);

[a]T

> exit;

Lecture 5 Semantics and Verification 2006

hm.cwb agent S = a.S1; agent S1 = b.0 + c.0; agent T = a.T1 + a.T2; agent T1 = b.0; agent T2 = c.0;