Lezione 21 MSC
HML: Hennessy-Milner Logic

Roberto Gorrieri

Syntax

Definition 5.1 The set $\mathcal{M}$ of *Hennessy-Milner formulae* over a set of actions **Act** is given by the following abstract syntax:

$$F, G ::= tt \mid ff \mid F \wedge G \mid F \vee G \mid \langle a \rangle F \mid [a]F,$$

where $a \in \mathsf{Act}$ and we use tt and ff to denote 'true' and 'false', respectively. If $A = \{a_1, \ldots, a_n\} \subseteq \mathsf{Act}$ $(n \geq 0)$, we use the abbreviation $\langle A \rangle F$ for the formula $\langle a_1 \rangle F \vee \cdots \vee \langle a_n \rangle F$ and $[A]F$ for the formula $[a_1]F \wedge \cdots \wedge [a_n]F$. (If $A = \emptyset$ then $\langle A \rangle F = ff$ and $[A]F = tt$.) ♦

Semantics

The semantics of formulae is defined with respect to a given LTS

$$(\mathsf{Proc}, \mathsf{Act}, \{\xrightarrow{a} \mid a \in \mathsf{Act}\}).$$

We shall use $[\![F]\!]$ to denote the set of processes in Proc that satisfy F. We now proceed to define this notation formally.

Definition 5.2 (Denotational semantics) We define $[\![F]\!] \subseteq \mathsf{Proc}$ for $F \in \mathcal{M}$ by

1. $[\![tt]\!] = \mathsf{Proc}$,
2. $[\![ff]\!] = \emptyset$,
3. $[\![F \wedge G]\!] = [\![F]\!] \cap [\![G]\!]$,
4. $[\![F \vee G]\!] = [\![F]\!] \cup [\![G]\!]$,
5. $[\![\langle a \rangle F]\!] = \langle \cdot a \cdot \rangle [\![F]\!]$,
6. $[\![[a]F]\!] = [\cdot a \cdot][\![F]\!]$,

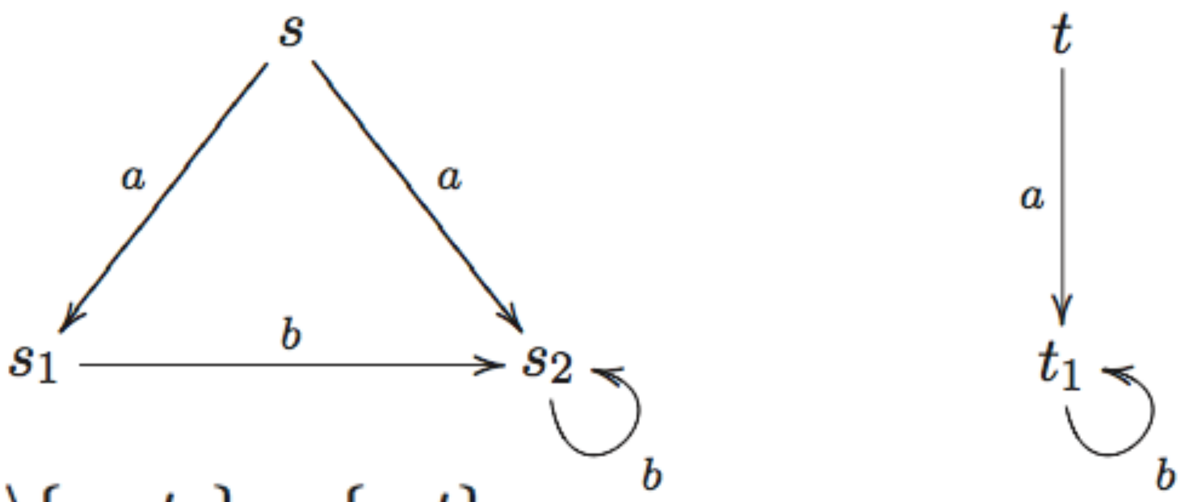
where we have used the set operators $\langle \cdot a \cdot \rangle, [\cdot a \cdot] : 2^{\mathsf{Proc}} \rightarrow 2^{\mathsf{Proc}}$ defined by

$$\langle \cdot a \cdot \rangle S = \{p \in \mathsf{Proc} \mid p \xrightarrow{a} p' \text{ and } p' \in S \text{ for some } p'\},$$
$$[\cdot a \cdot] S = \{p \in \mathsf{Proc} \mid p \xrightarrow{a} p' \text{ implies } p' \in S \text{ for each } p'\}.$$

We write $p \models F$, read as 'p satisfies F', iff $p \in [\![F]\!]$.

Two formulae are *equivalent* iff they are satisfied by the same processes in every transition system. ♦

Example (1)



$$\langle \cdot a \cdot \rangle \{s_1, t_1\} = \{s, t\}$$

$$[\cdot a \cdot]\{s_1, t_1\} = \{p \in \mathsf{Proc} \mid p \xrightarrow{a} p' \text{ implies } p' \in \{s_1, t_1\} \text{ for each } p'\}$$

$$[\cdot a \cdot]\{s_1, t_1\} = \{s_1, s_2, t, t_1\}$$

- Exercise: compute $\langle \cdot b \cdot \rangle \{s_1, t_1\}$ and $[\cdot b \cdot]\{s_1, t_1\}$

Example (2)

$$
\begin{aligned}
[\![\langle \text{coffee} \rangle tt]\!] &= \langle \cdot \text{coffee} \cdot \rangle [\![tt]\!] \\
&= \langle \cdot \text{coffee} \cdot \rangle \textsf{Proc} \\
&= \{P \mid P \stackrel{\text{coffee}}{\rightarrow} P' \text{ for some } P' \in \textsf{Proc}\} \\
&= \{P \mid P \stackrel{\text{coffee}}{\rightarrow}\}.
\end{aligned}
$$

$$
\begin{aligned}
[\![[\text{tea}] ff]\!] &= [\cdot \text{tea} \cdot][\![ff]\!] \\
&= [\cdot \text{tea} \cdot]\emptyset \\
&= \{P \mid P \stackrel{\text{tea}}{\rightarrow} P' \text{ implies } P' \in \emptyset \text{ for each } P'\} \\
&= \{P \mid P \stackrel{\text{tea}}{\nrightarrow}\}.
\end{aligned}
$$

P ⊨ <a>tt means "P can do a now"

P ⊨ [a]ff means "P cannot do a now"

Exercise (1)

- What is the meaning of the following formulae?
- [coffee]<biscuit>tt
- <coffee>tt ∨ <tea>tt
- <coffee>tt ∧ [tea]ff
- [coffee][coffee]<tea>tt
- [coffee][tea]ff
- What about <a>ff and [a]tt ?

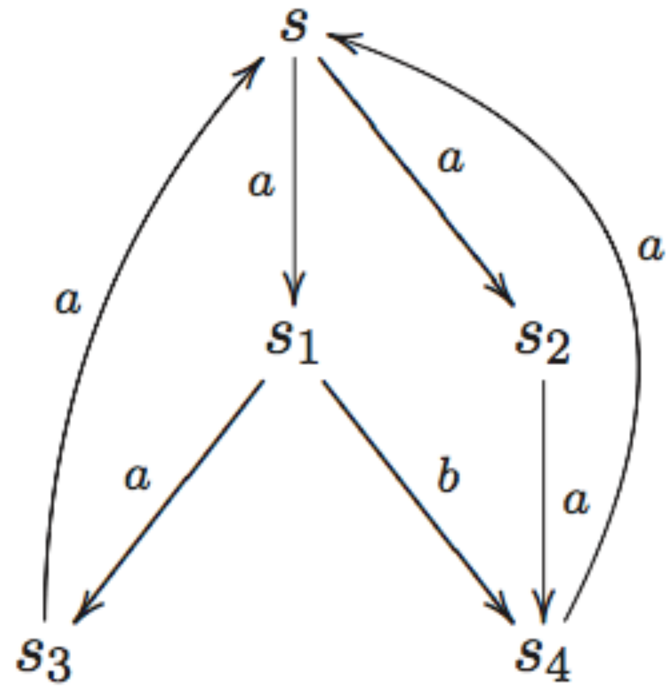
  answer: <a>ff equivalent to ff while [a]tt to tt

Exercise (2)

- Find a formula F such that a.b + a.c satisfies F, while a.(b + c) does not satisfy F
  - Answer: F = <a>[b]ff
- Find a formula G such that a.(b.c + b.d) satisfies G, but a.b.c + a.b.d doesn’t
  - Answer: G = [a]<b><c>tt
- Find a formula H such that a.(b.c +b.d) + a.b.d satisfies H, but a.(b.c +b.d) doesn’t
  - Answer: H = <a>[b]<d>tt

Exercise (3)

- Which of the following hold?



$$s \stackrel{?}{\models} [a]\langle b\rangle \mathit{tt},$$

$$s \stackrel{?}{\models} \langle a\rangle\langle b\rangle \mathit{tt},$$

$$s \stackrel{?}{\models} [a]\langle a\rangle[a][b]\mathit{ff},$$

$$s \stackrel{?}{\models} \langle a\rangle\big(\langle a\rangle \mathit{tt} \wedge \langle b\rangle \mathit{tt}\big),$$

$$s \stackrel{?}{\models} [a]\big(\langle a\rangle \mathit{tt} \vee \langle b\rangle \mathit{tt}\big),$$

$$s \stackrel{?}{\models} \langle a\rangle\big([b][a]\mathit{ff} \wedge \langle b\rangle \mathit{tt}\big),$$

$$s \stackrel{?}{\models} \langle a\rangle\big([a](\langle a\rangle \mathit{tt} \wedge [b]\mathit{ff}) \wedge \langle b\rangle \mathit{ff}\big).$$

$$s \stackrel{?}{\models} \langle a\rangle \mathit{tt},$$

$$s \stackrel{?}{\models} \langle b\rangle \mathit{tt},$$

$$s \stackrel{?}{\models} [a]\mathit{ff},$$

$$s \stackrel{?}{\models} [b]\mathit{ff},$$

Exercise (4)

- Considering the lts of the previous slide:

2. *Compute the following sets using the denotational semantics for Hennessy–Milner logic.*

$$
\begin{aligned}
[\![[a][b]\mathit{ff}]\!] &= ?, \\
[\![\langle a\rangle\big(\langle a\rangle\mathit{tt} \wedge \langle b\rangle\mathit{tt}\big)]\!] &= ?, \\
[\![[a][a][b]\mathit{ff}]\!] &= ?, \\
[\![[a]\big(\langle a\rangle\mathit{tt} \vee \langle b\rangle\mathit{tt}\big)]\!] &= ?.
\end{aligned}
$$

Exercise 5.7 *Find an LTS with initial state s that satisfies all the following properties:*

$$
\begin{aligned}
&\langle a\rangle(\langle b\rangle\langle c\rangle\mathit{tt} \wedge \langle c\rangle\mathit{tt}), \\
&\langle a\rangle\langle b\rangle([a]\mathit{ff} \wedge [b]\mathit{ff} \wedge [c]\mathit{ff}), \\
&[a]\langle b\rangle([c]\mathit{ff} \wedge \langle a\rangle\mathit{tt}).
\end{aligned}
$$

Exercise (5)

Exercise 5.4 *Consider an everlasting clock whose behaviour is defined thus:*

$$\mathrm{Clock} \stackrel{\mathrm{def}}{=} \mathrm{tick}.\mathrm{Clock}.$$

Prove that the process Clock *satisfies the formula*

$$[\mathrm{tick}](\langle \mathrm{tick}\rangle tt \wedge [\mathrm{tock}] ff).$$

Show also that, for each $n \geq 0$, *the process* Clock *satisfies the formula*

$$\underbrace{\langle \mathrm{tick}\rangle \cdots \langle \mathrm{tick}\rangle}_{n \text{ times}} tt.$$

♦

Satisfaction relation

- Alternative, direct, inductive definition:

$$
\begin{aligned}
&P \models tt \text{ for each } P,\\
&P \models ff \text{ for no } P,\\
&P \models F \wedge G \text{ iff } P \models F \text{ and } P \models G,\\
&P \models F \vee G \text{ iff } P \models F \text{ or } P \models G,\\
&P \models \langle a \rangle F \text{ iff } P \xrightarrow{a} P' \text{ for some } P' \text{ such that } P' \models F,\\
&P \models [a]F \text{ iff } P' \models F \text{ for each } P' \text{ such that } P \xrightarrow{a} P'.
\end{aligned}
$$

Exercise 5.6 *Show that the above definition of the satisfaction relation is equivalent to that given in Definition 5.2. Hint: Use induction on the structure of formulae.* ♦

Negation

Note that logical negation is *not* one of the constructs in the abstract syntax for $\mathcal{M}$. However, the language $\mathcal{M}$ *is* closed under negation in the sense that, for each formula $F \in \mathcal{M}$, there is a formula $F^c \in \mathcal{M}$ that is equivalent to the negation of F. This formula F^c is defined inductively on the structure of F as follows.

1. $tt^c = ff$,
2. $ff^c = tt$,
3. $(F \wedge G)^c = F^c \vee G^c$,
4. $(F \vee G)^c = F^c \wedge G^c$,
5. $(\langle a \rangle F)^c = [a]F^c$,
6. $([a]F)^c = \langle a \rangle F^c$.

Note, for instance, that

$$(\langle a \rangle tt)^c = [a] ff,$$
$$([a] ff)^c = \langle a \rangle tt.$$

Proposition 5.1 Let $(\mathsf{Proc}, \mathsf{Act}, \{\xrightarrow{a} \mid a \in \mathsf{Act}\})$ be an LTS. Then, for every formula $F \in \mathcal{M}$, it holds that $[\![F^c]\!] = \mathsf{Proc} \setminus [\![F]\!]$.

Proof. The proposition can be proved by structural induction on F. The details are left as an exercise to the reader. $\square$

Bisimilarity and satisfiability

- A = a.A + a.0   B = a.a.B + a.0
- A is not bisimilar to B: Why? How to match transition A –a-> A?
- Is there any formula distinguishing A and B?

What about F = <a><a>[a]ff ? Check that A satisfies F, but B does not.

- In general, are two not bisimilar processes distinguishable by a HML formula? And do two bisimilar processes satisfy the same formulae?

Hennessy-Milner theorem

Let (Proc, Act, →) be an image-finite lts. Then, for all P and Q in Proc:

P ~ Q iff (P ⊨ F iff Q ⊨ F for all F)

i.e., P and Q are bisimilar iff they satisfy the same set of formulae.

Proof: ➔) (No need of the assumption of image-finiteness)

Assume P ~ Q and P ⊨ F. We want to prove that Q ⊨ F by structural induction on F. (By symmetry, this is enough to establish the thesis.)

Proof (1)

- $F = tt$ By definition of $\models$, $Q \models tt$
- $F = ff$ Impossible as P does not satisfy ff
- $F = F_1 \wedge F_2$ The inductive hypothesis is
  " $R \sim S$ and $R \models F_i$ implies $S \models F_i$ $i = 1, 2$"

We know that $P \sim Q$ and $P \models F_1 \wedge F_2$. By definition of $\models$, $P \models F_i$ for $i = 1, 2$. By applying induction, $Q \models F_i$ $i = 1, 2$.

Then, $Q \models F_1 \wedge F_2$ by definition of $\models$.

- $F = F_1 \vee F_2$ analogous
- $F = \langle a \rangle G$ The inductive hypothesis is
  " $R \sim S$ and $R \models G$ implies $S \models G$"

We know that $P \sim Q$ and $P \models \langle a \rangle G$. By definition of $\models$, there exists P' such that $P \xrightarrow{a} P'$ and $P' \models G$. By definition of $\sim$, there exists Q' such that $Q \xrightarrow{a} Q'$ and $P' \sim Q'$. Then, by inductive hypothesis, $Q' \models G$ and so, by def of $\models$, $Q \models \langle a \rangle G$.

Proof (2)

- F = [a]G The inductive hypothesis is

  " R ~ S and R ⊨ G implies S ⊨ G"

We know that P ~ Q and P ⊨ [a]G . By definition of ⊨, for all P' such that P—a->P', P' ⊨ G holds. By definition of ~, for each

Q—a->Q' there must exist a P' such that P—a->P' with P' ~ Q'. Then, by induction, Q' ⊨ G for all Q' that derive with a. Then, by def of ⊨, Q ⊨ [a]G .

🡐) (P and Q satify the same formulae implies P ~ Q) We need the hypothesis of image-finiteness. We prove that relation

R = {(P, Q) | P and Q satisfy the same formulae}

is a bisimulation.

Assume (P, Q) in R and P –a->P'. We will prove that there exists Q' such that Q—a->Q' with (P', Q') in R. Since R is symmetric, this is enough for proving that R is a bisimulation.

Proof (3)

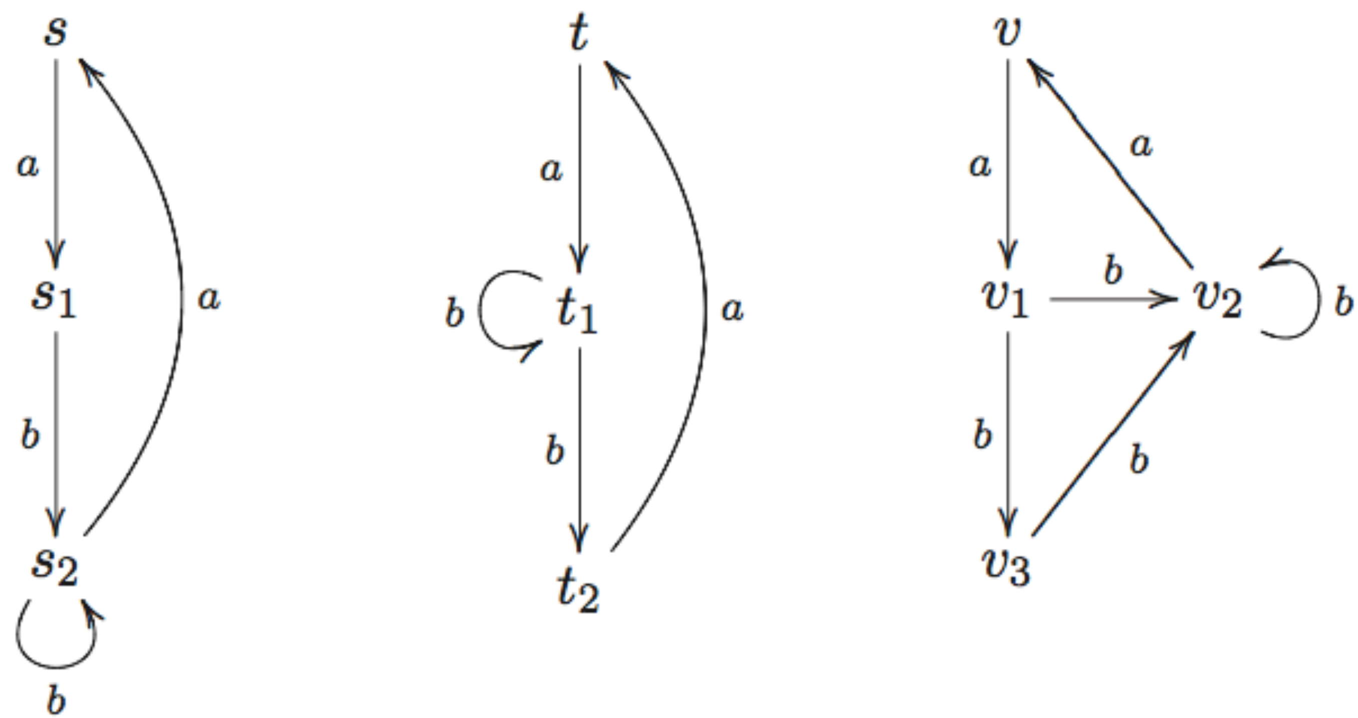
Let us assume, towards a contradiction, that there is no Q' such that Q—a->Q' and (P', Q') in R. Since the LTS is image-finite, the set {Q' | Q—a->Q'} is finite; let denote it by $\{Q_1, Q_2, \ldots, Q_n\}$ with n≥0. Since, by assumption, (P', Q') not in R for Q' in $\{Q_1, Q_2, \ldots, Q_n\}$, it follows that none of the Q_i satisfies the same formulae of P'. Then for each i = 1, 2, .., n, there exists a formula F_i such that $P' \models F_i$ while Q_i does not satisfy F_i. Therefore, the formula

$$F = \langle a \rangle (F_1 \wedge F_2 \ \ldots \wedge F_n)$$

is such that $P \models F$ while Q does not satify F. Hence, contradiction: (P, Q) not in R. Hence, the hypothesis that there is no Q' such that Q—a->Q' and (P', Q') in R is wrong, and, in conclusion, R is a bisimulation.

(If n = 0, then F = <a>tt and P—a->P' while Q cannot do a)

Exercise 5.10 *Consider the following LTS:*



Argue that $s \not\sim t$, $s \not\sim v$ and $t \not\sim v$. Next, find a distinguishing formula of Hennessy–Milner logic for each of the pairs

$$s \text{ and } t, \quad s \text{ and } v, \quad t \text{ and } v.$$

Verify your claims in the Edinburgh Concurrency Workbench (use the `strongeq` *and* `checkprop` *commands) and check whether you have found the shortest distinguishing formula (use the* `dfstrong` *command).* ♦

Solution

- s and t not bisimilar

F = [a][b]<a>tt   is satisfied only by s

- s and v not bisimilar

F above is satified only by s

- t and v not bisimilar

G = [a][b]<b>tt is satisfied by v only

Exercise

Exercise 5.11 *For each of the following pairs of CCS expressions, decide whether they are strongly bisimilar and, if they are not, find a distinguishing formula in Hennessy–Milner logic:*

$$\begin{array}{rcl} b.a.\mathbf{0} + b.\mathbf{0} & \textit{and} & b.(a.\mathbf{0} + b.\mathbf{0}); \\ a.(b.c.\mathbf{0} + b.d.\mathbf{0}) & \textit{and} & a.b.c.\mathbf{0} + a.b.d.\mathbf{0}; \\ a.\mathbf{0} \mid b.\mathbf{0} & \textit{and} & a.b.\mathbf{0} + b.a.\mathbf{0}; \;\; \textit{and} \\ (a.\mathbf{0} \mid b.\mathbf{0}) + c.a.\mathbf{0} & \textit{and} & a.\mathbf{0} \mid (b.\mathbf{0} + c.\mathbf{0}). \end{array}$$

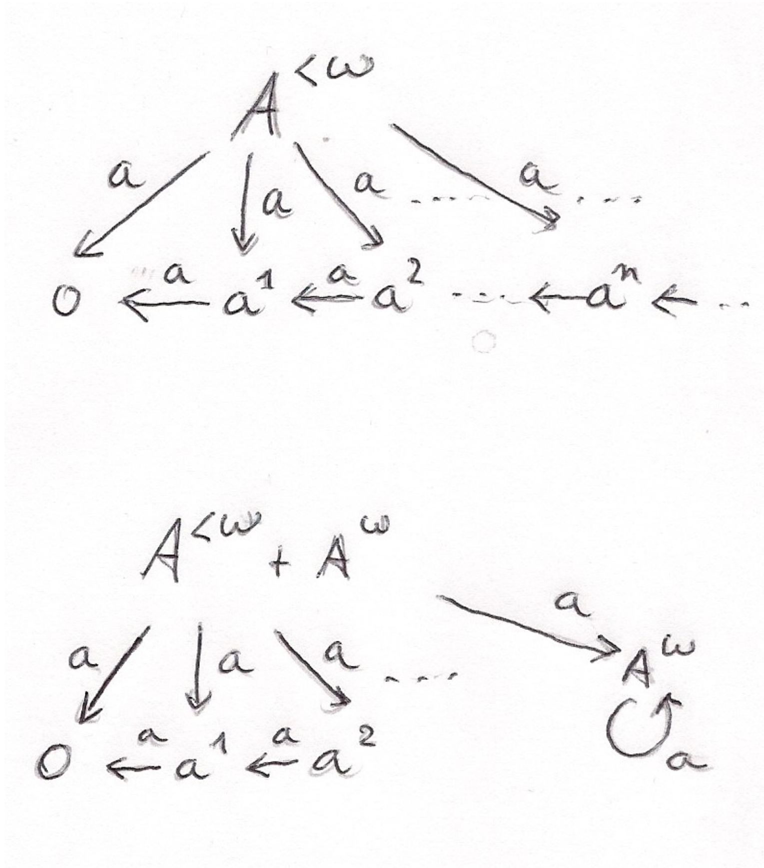
Verify your claims in the Edinburgh Concurrency Workbench (use the `strongeq` *and* `checkprop` *commands) and check whether you have found the shortest distinguishing formula (use the* `dfstrong` *command).* ♦

Solution

- b.a + b   and   b.(a + b)   F = [b]<b>tt
- a.(b.c + b.d)   and a.b.c + a.b.d G = [a]<b><c>tt
- a |b is bisimilar to a.b +b.a
- (a | b) + c.a   and   a | (b + c)   H = <a><c>tt

Why image-finite?

$A^{<w} = \sum_{i \text{ in } N} a^i$
$a^0 = 0 \quad a^{i+1} = a.a^i$
$A^w = a.A^w$
$A^{<w}$ and $A^{<w} + A^w$ not bisimilar
but they satisfy the same formulae



$A^w \vDash F$ iff $a^i \vDash F$ where i=md(F)

Modal depth:
md(tt) = md(ff) = 0
$md(F_1 \wedge F_2) = md(F_1 \vee F_2) = \max\{md(F_1), md(F_2)\}$
md(<a>F) = md([a]F) = 1 + md(F)