Situation Calculus-based Online Plan Recognition in Continuous Domains

Christoph Schwering

RWTH Aachen University

December 20, 2011
Motivation
Approach

Model

```plaintext
proc overtake(V, W)
  behind(V, W)?;
  leftLaneChange(V);
  wait for behind(W, V);
  rightLaneChange(V)
...
```

Interpretor

Observations

- at time 0: $\text{pos}(A) = (10, -2)$
- at time 1: $\text{pos}(A) = (25, -2)$
- at time 2: $\text{pos}(A) = (40, 0)$
...

Set of programs that explain the observations.
Outline

Introduction
  Related Work
  Modeling

Semantics
  Time and Continuous Change
  Multiple Agents
  Robustness

Plan Recognition by Program Execution
  observe Actions
  Online Heuristic

Evaluation

Discussion
## Related Work

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tools</td>
<td>circumscription</td>
<td>Bayesian network</td>
<td>situation calculus</td>
</tr>
<tr>
<td>Modeling</td>
<td>first-order logic</td>
<td>ConGolog</td>
<td></td>
</tr>
<tr>
<td>Focus</td>
<td>abstraction</td>
<td>abstraction, online</td>
<td></td>
</tr>
<tr>
<td>Observations</td>
<td>primitive action occurrences</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>likely</td>
<td>likely</td>
<td>consistent/likely</td>
</tr>
<tr>
<td>HMM</td>
<td>HMM</td>
<td>planner</td>
</tr>
<tr>
<td>hierarchical MDPs</td>
<td>plan tree grammars</td>
<td>STRIPS, goal library</td>
</tr>
<tr>
<td>abstraction, uncertainty</td>
<td>abstraction, partial ordering</td>
<td>–</td>
</tr>
</tbody>
</table>

primitive action occurrences
Modeling

- Global cartesian view
- Vehicle = rectangle
- Instantaneous actions $setYaw$, $setVelocity$
Modeling

- Global cartesian view
- Vehicle = rectangle
- Instantaneous actions $setYaw$, $setVelocity$
Modeling

- Global cartesian view
- Vehicle = rectangle
- Instantaneous actions \( \text{setYaw}, \text{setVelocity} \)
Modeling

- Global cartesian view
- Vehicle = rectangle
- Instantaneous actions $set\text{Y}\text{aw}$, $set\text{Veloc}$
Modeling

- Global cartesian view
- Vehicle = rectangle
- Instantaneous actions $setYaw$, $setVeloc$
Global cartesian view

Vehicle = rectangle

Instantaneous actions setYaw, setVelocity
Programs

\[
\text{proc } \text{leftLaneChange}(V) \\
\text{pick } \gamma \in \{4^\circ, 6^\circ, \ldots, 12^\circ\} \text{ do} \\
\text{setYaw}(V, \gamma) \\
\text{endpick;} \\
\text{onRightLane}(V) \?; \\
\% \text{ time passes indefinitely} \\
\text{setYaw}(V, 0^\circ); \\
\text{onLeftLane}(V) \? \\
\text{endproc}
\]
Programs

\[
\begin{align*}
\text{proc } & \text{leftLaneChange}(V) \\
\text{pick } & \gamma \in \{4^\circ, 6^\circ, \ldots, 12^\circ\} \text{ do} \\
& \text{setYaw}(V, \gamma) \\
\text{endpick;} \\
& \text{onRightLane}(V) \\
\% & \text{time passes indefinitely} \\
& \text{setYaw}(V, 0^\circ) \\
& \text{onLeftLane}(V) \\
\text{endproc}
\end{align*}
\]
**Programs**

**proc** `leftLaneChange(V)`

`pick` $\gamma \in \{4^\circ, 6^\circ, \ldots, 12^\circ\}$ `do`

`setYaw(V, \gamma)`

`endpick;`

`onRightLane(V) ?;`

`% time passes indefinitely`

`setYaw(V, 0^\circ);`

`onLeftLane(V) ?`

`endproc`
Programs

\[
\text{proc } \text{leftLaneChange}(V) \\
\quad \text{pick } \gamma \in \{4^\circ, 6^\circ, \ldots, 12^\circ\} \quad \text{do} \\
\quad \quad \text{setYaw}(V, \gamma) \\
\quad \text{endpick}; \\
\quad \text{onRightLane}(V) \quad \text{?};; \\
\quad \text{% time passes indefinitely} \\
\quad \text{setYaw}(V, 0^\circ); \\
\quad \text{onLeftLane}(V) \quad \text{?} \\
\text{endproc}
\]
Semantics

What is needed to make it work?

Golog $Trans +$

- Flexible timing
- Continuous change
- Multi-agent
- Robustness
Time and Continuous Change

different points in time
Time and Continuous Change

From temporal sequential Golog:

\[ \text{time}(A(\vec{x}, \tau)) = \tau \]

\[ \text{start}(\text{do}(a, s)) = \text{time}(a) \]

From cc-Golog:

\[ \phi[s, \tau] \quad \text{evaluate } \phi \text{ in } s \text{ at time } \tau \]

\[ \alpha[s, \tau] \quad \text{append new time parameter} \]

\[ \text{e.g. } \text{jump}[s, \tau] = \text{jump}(\tau) \]
Time and Continuous Change

primitive action

\[\text{Trans}(\alpha, s, \delta, s') \equiv \delta = \text{Nil} \land \]
\[\exists \tau . \tau \geq \text{start}(s) \land \]
\[\text{Poss}(\alpha[s, \tau], s) \land \]
\[s' = \text{do}(\alpha[s, \tau], s)\]
Time and Continuous Change

\[
Trans(\alpha, s, \delta, s') \equiv \delta = \text{Nil} \land \exists \tau. \tau \geq \text{start}(s) \land Poss(\alpha[s, \tau], s) \land s' = do(\alpha[s, \tau], s)
\]
Time and Continuous Change

\[ \text{primitive action} \]

\[ \text{Trans}(\alpha, s, \delta, s') \equiv \delta = \text{Nil} \land \exists \tau. \tau \geq \text{start}(s) \land \]

\[ \text{Poss}(\alpha[s, \tau], s) \land \]

\[ s' = \text{do}(\alpha[s, \tau], s) \]

\[ \text{monotonicity} \]

\[ \text{constrains } \tau \text{ further} \]
Time and Continuous Change

\[ Trans(\alpha, s, \delta, s') \equiv \delta = Nil \land \exists \tau. \tau \geq start(s) \land Poss(\alpha[s, \tau], s) \land s' = do(\alpha[s, \tau], s) \]

primitive action

monotonicity

constrains \( \tau \) further

advance to time \( \tau \)
**Time and Continuous Change**

\[
Trans(\alpha, s, \delta, s') \equiv \delta = Nil \land \\
\exists \tau. \tau \geq start(s) \land \\
Poss(\alpha[s, \tau], s) \land \\
s' = do(\alpha[s, \tau], s)
\]

\[
Poss(waitFor(\phi, \tau), s) \equiv \phi[s, \tau]
\]
Multi-agent: \( \sigma_1 \parallel \ldots \parallel \sigma_n \) explains observations?

actor 1

actor \( n \)
Multiple Agents

Multi-agent: $\sigma_1 \parallel \ldots \parallel \sigma_n$ explains observations?

actor 1  actor n

Concurrency as in ConGolog:

\[
Trans(\sigma_1 \parallel \sigma_2, s, \delta, s') \equiv \exists \delta'. Trans(\sigma_1, s, \delta', s') \land \delta = \delta' \parallel \sigma_2 \lor \\
\exists \delta'. Trans(\sigma_2, s, \delta', s') \land \delta = \sigma_1 \parallel \delta'
\]
Robustness

Observed trace
Robustness

Observed trace + model trace

Hypothesis: driving straight?
Robustness

Observed trace + model trace + lateral tolerance

Hypothesis: driving straight?
Robustness

Observed trace + model trace + lateral tolerances

Hypothesis: driving straight?
**Robustness**

*Observed trace* + *model trace* + *weighted lateral tolerances*

Hypothesis: driving straight?  **Likely**
Robustness

**Observed trace** + **model trace** + weighted lateral tolerances

Hypothesis: driving straight? **Less likely**
Robustness

Observed trace + model trace + weighted lateral tolerances

Hypothesis: driving straight? Unlikely
Robustness

- Tolerances by stochastic actions

\( \text{Choice}(\beta, \alpha) \text{ and } \text{prob}_0(\beta, \alpha, s) \mapsto [0, 1] \)
Robustness

- Tolerances by stochastic actions
  \( \text{Choice}(\beta, \alpha) \) and \( \text{prob}_0(\beta, \alpha, s) \mapsto [0, 1] \)
- Rate situation by reward
  \( r(s) \mapsto \mathbb{R} \)
Robustness

- Tolerances by stochastic actions
  \( Choice(\beta, \alpha) \) and \( prob_0(\beta, \alpha, s) \mapsto [0, 1] \)

- Rate situation by reward
  \( r(s) \mapsto \mathbb{R} \)

- Nondeterminism \( \rightarrow \) choose best alternative
Robustness

- Tolerances by stochastic actions
  \( \text{Choice}(\beta, \alpha) \text{ and } \text{prob}_0(\beta, \alpha, s) \mapsto [0, 1] \)

- Rate situation by reward
  \[ r(s) \mapsto \mathbb{R} \]

- Nondeterminism → choose best alternative:
  1. Decompose \( \sigma \) into \((\gamma; \delta)\) atomic action
  2. Find best \((\gamma; \delta)\) amongst all decompositions
  3. Execute \( \gamma \)
Robustness

▶ Tolerances by stochastic actions
Choice(\(\beta, \alpha\)) and prob_0(\(\beta, \alpha, s\)) \(\mapsto [0, 1]\)

▶ Rate situation by reward
\(r(s) \mapsto \mathbb{R}\)

▶ Rate program by estimated reward
\(value(r, \sigma, s) \mapsto \mathbb{R}\)

▶ Nondeterminism \(\rightarrow\) choose best alternative:
1. Decompose \(\sigma\) into \((\gamma; \delta)\)
atomic action
2. Find best \((\gamma; \delta)\) amongst all decompositions
3. Execute \(\gamma\)
Robustness: Decomposition

Next($\sigma, \gamma, \delta$)

input \quad next atomic \quad remainder
Robustness: Decomposition

Like $Trans$ without execution, e.g.:

$$Next(\alpha, \gamma, \delta) \equiv \gamma = \alpha \land \delta = Nil$$

$$Next(\sigma_1 | \sigma_2, \gamma, \delta) \equiv Next(\sigma_1, \gamma, \delta) \lor Next(\sigma_2, \gamma, \delta)$$
Robustness: Transition

\[ \text{transPr}(r, \sigma, s, \delta, s') = p \equiv \]
\[
\begin{align*}
\text{if } & \exists^1 \gamma_1, \delta_1 . \text{Next}(\sigma, \gamma_1, \delta_1) \land \\
& (\forall \gamma_2, \delta_2 . \text{Next}(\sigma, \gamma_2, \delta_2) \supset \text{decomposition } \gamma_1; \delta_1 \text{ is optimal} \\
& \quad \text{value}(r, (\gamma_1; \delta_1), s) \geq \text{value}(r, (\gamma_2; \delta_2), s) \\
\text{then } & (\text{if } \delta = \delta_1 \text{ then } p = \text{transAtPr}(r, \gamma_1, \delta_1, s, s') \text{ else } p = 0) \\
\text{else } & p = 0
\end{align*}
\]

execute \( \gamma_1 \)
Robustness

Why decomposition? Decision theory + concurrency

\( Trans \) recursively follows syntax tree
\( \leadsto \) does not know “what comes after”
Robustness

Why decomposition? Decision theory + concurrency

Trans recursively follows syntax tree
\]$ does not know “what comes after”

Program decomposition
\]$ full remaining program is always known
\]$ can resolve nondeterminism with remainder in mind
Robustness: Atomic Complex Actions

\[ \text{atomic}(a; b) \parallel c \not\sim do([a, c, b], S_0) \]
\[ \not\sim do([a, b, c], S_0) \]
\[ \not\sim do([c, a, b], S_0) \]
Plan Recognition by Program Execution

Plan recognition...

- as satisfiability
- by iterative filtering of allConsistPlans
- by program execution
Plan Recognition by Program Execution

Plan recognition...

- as satisfiability
- by iterative filtering of \textit{allConsistPlans}
- by program execution
**observe** *Actions*

\[ \text{Poss}(\text{observe}(\tau, \phi, \tau'), s) \equiv \tau = \tau' \land \phi[s, \tau] \]

Execution of \text{observe}(\tau, \phi) means \phi was observed at time \tau
**observe** Actions

\[ \text{Poss}(\text{observe}(\tau, \phi, \tau'), s) \equiv \tau = \tau' \land \phi[s, \tau] \]

Execution of \text{observe}(\tau, \phi) means \( \phi \) was observed at time \( \tau \)

\((\text{observe}(\tau_1, \phi_1); \ldots; \text{observe}(\tau_n, \phi_n))\)
**observe** Actions

\[ \text{Poss}(\text{observe}(\tau, \phi, \tau'), s) \equiv \tau = \tau' \land \phi[s, \tau] \]

Execution of \( \text{observe}(\tau, \phi) \) means \( \phi \) was observed at time \( \tau \)

\[ \sigma \quad (\text{observe}(\tau_1, \phi_1); \ldots; \text{observe}(\tau_n, \phi_n)) \]
 observes Actions

\[
\text{Poss}(\text{observe}(\tau, \phi, \tau'), s) \equiv \tau = \tau' \land \phi[s, \tau]
\]

Execution of \text{observe}(\tau, \phi) means \phi was observed at time \tau

\[
\sigma \parallel (\text{observe}(\tau_1, \phi_1); \ldots; \text{observe}(\tau_n, \phi_n))
\]
Online Heuristic

1. New observation \((\tau, \phi)\) present:

\[
\delta' = \delta \parallel observe(\tau, \phi)
\]

merge observation
Online Heuristic

1. New observation \((\tau, \phi)\) present:

\[\delta' = \delta \parallel observe(\tau, \phi)\]

2. Enough \textit{observe} actions buffered:

\[p' = p \cdot transPr(r, \tilde{\delta}, s, \tilde{\delta}', s')\]
Online Heuristic

1. New observation \((\tau, \phi)\) present:

\[
\delta' = \delta \parallel observe(\tau, \phi)
\]

2. Enough \emph{observe} actions buffered:

\[
p' = p \cdot transPr(r, \delta, s, \delta', s')
\]

3. Reiterate.

merge observation

resolves nondeterminism
**Approach Summary**

**Model**

```
proc overtake(V, W)
  behind(V, W)?;
  leftLaneChange(V);
  wait for behind(W, V);
  rightLaneChange(V)
...
```

**Observations**

- at time 0: \( pos(A) = (10, -2) \)
- at time 1: \( pos(A) = (25, -2) \)
- at time 2: \( pos(A) = (40, 0) \)
  ...

Set of programs that explain the observations.
Approach Summary

Candidate programs of the form
\[ \sigma_1 \parallel \ldots \parallel \sigma_n \]
for \( n \) actors.

Model

```plaintext
proc overtake(V, W)
    behind(V, W)?;
    leftLaneChange(V);
    wait for behind(W, V);
    rightLaneChange(V)
    ...
```

Observation Program \( \theta \)

\[ \text{obs.}(0, \text{pos}(A) = (10, -2)) \parallel \]
\[ \text{obs.}(1, \text{pos}(A) = (25, -2)) \parallel \]
\[ \text{obs.}(2, \text{pos}(A) = (40, 0)) \parallel \]
\[ \ldots \]

Set of programs that explain the observations.

Observations

- at time 0: \( \text{pos}(A) = (10, -2) \)
- at time 1: \( \text{pos}(A) = (25, -2) \)
- at time 2: \( \text{pos}(A) = (40, 0) \)
- \( \ldots \)
Candidate programs of the form \( \sigma_1 \parallel \ldots \parallel \sigma_n \) for \( n \) actors.

Observation Program \( \theta \)

\[
\text{obs.}(0, \text{pos}(A) = (10, -2)) \parallel \\
\text{obs.}(1, \text{pos}(A) = (25, -2)) \parallel \\
\text{obs.}(2, \text{pos}(A) = (40, 0)) \parallel \\
\ldots
\]

Set of programs with confidences

Model

\[
\text{proc} \ \text{overtake}(V, W) \\
\begin{align*}
&\text{behind}(V, W)?; \\
&\text{leftLaneChange}(V); \\
&\text{wait for} \ \text{behind}(W, V); \\
&\text{rightLaneChange}(V) \\
\ldots
\end{align*}
\]
Evaluation

- Prototype in ECLiPSe-CLP
- Sampling
- Linear constraint solver
  for equations from \textit{waitFor}, \textit{observe}
Demo

Video #1  Video #2
Accomplishments

✓ Flexible timing
✓ Continuous change
✓ Multi-agent
✓ Robustness
  Model simplifies world
Conclusion

Plan Recognition by Program Execution

Accomplishments

✓ Flexible timing
✓ Continuous change
✓ Multi-agent
✓ Robustness
  Model simplifies world
  Sensor noise

Features

➤ Keeps it simple
➤ Sensor noise
➤ Efficient
Future Work

- Nonlinear constraints
- Extrapolate situation + remaining program
Appendix


The axiomizer must guarantee:

\[ D \models \text{Choice}(\beta, \alpha) \land (\exists \tau . \tau \geq \text{start}(s) \land \text{Poss}(\alpha[s, \tau], s)) \supset \text{prob}_0(\beta, \alpha, s) > 0 \]

\[ D \models (\exists \alpha . \text{Choice}(\beta, \alpha) \land \exists \tau . \tau \geq \text{start}(s) \land \text{Poss}(\alpha[s, \tau], s)) \supset \sum_{\{\alpha | \text{Choice}(\beta, \alpha) \land \exists \tau . \tau \geq \text{start}(s) \land \text{Poss}(\alpha[s, \tau], s)\}} \text{prob}_0(\beta, \alpha, s) = 1 \]

\[ D \models \forall \beta . \exists f . \forall \alpha . \text{Choice}(\beta, \alpha) \supset (\exists i) f(i) = \alpha \]
Robustness: value

\[ \text{value}(r, \sigma, S_0) \geq 3 \frac{1}{3} \]
Robustness: $value$

$value(r, \sigma, S_0) \geq 3$

Formulas

$S_0, r = 3\frac{1}{3}$

- $do(a_{11}, S_0), r = 1$
  - $p = \frac{1}{2}$
  - $do([a_{11}, a_{21}], S_0), r = 3$
  - $p = \frac{1}{2}$
  - $do([a_{11}, a_{22}], S_0), r = 2$

- $do(a_{12}, S_0), r = 4$
  - $p = \frac{1}{2}$
  - $do([a_{12}, a_{21}], S_0), r = 10$
  - $p = \frac{1}{2}$
  - $do([a_{12}, a_{22}], S_0), r = 0$

- $do(a_{13}, S_0), r = 4$
  - $p = \frac{1}{2}$
  - $do([a_{13}, a_{21}], S_0), r = 2$
  - $p = \frac{1}{2}$
  - $do([a_{13}, a_{22}], S_0), r = 2$
Robustness: \( value \)

\[
value(r, \sigma, S_0) \geq 3\frac{1}{6}
\]

\[\begin{align*}
S_0, r = 3\frac{1}{3} & \quad \text{do}(a_{11}, S_0), r = 1 \\
& \quad \quad \quad \quad \text{do}([a_{11}, a_{21}], S_0), r = 3 \\
& \quad \quad \quad \quad \text{do}([a_{11}, a_{22}], S_0), r = 2 \\
& \quad \quad \quad \quad \text{do}(a_{12}, S_0), r = 4 \\
& \quad \quad \quad \quad \text{do}([a_{12}, a_{21}], S_0), r = 10 \\
& \quad \quad \quad \quad \text{do}([a_{12}, a_{22}], S_0), r = 0 \\
& \quad \quad \quad \quad \text{do}(a_{13}, S_0), r = 4 \\
& \quad \quad \quad \quad \text{do}([a_{13}, a_{21}], S_0), r = 2 \\
& \quad \quad \quad \quad \text{do}([a_{13}, a_{22}], S_0), r = 2
\end{align*}\]
Robustness: \textit{value}

\[ \text{value}(r, \sigma, S_0) = 3\frac{5}{6} \]
Robustness: \textit{value}

\[
Best(r, \sigma, s) \overset{\text{def}}{=} \forall P. \left( \forall s', s''. P(s') \land P(s'') \supset s' \not\subset s'' \right) \supset \\
\sum \{ (p, s') | \exists \delta. \, \text{transPr}^*(r, \sigma, s, \delta, s') = p \land \\
p > 0 \land P(s') \} \cdot p \cdot r(s') \leq r(s)
\]

\[
value(r, \sigma, s) \overset{\text{def}}{=} \sum \{ (p, s') | \exists \delta. \, \text{transPr}^*(r, \sigma, s, \delta, s') = p \land \\
p > 0 \land Best(r, \delta, s') \land \\
\neg \exists s'', \delta. \, \text{transPr}^*(r, \sigma, s, \delta, s'') > 0 \land \\
Best(r, \delta, s'') \land s'' \subset s' \} \cdot p \cdot r(s')
\]
Robustness: Sum Axiomatization

\[ \sum_{\{\vec{x} \mid \Phi[\vec{X}/\vec{x}]\}} \nu(\vec{x}) \]

\[ \text{sum}_\nu(\Phi(\vec{X})) = \nu \stackrel{\text{def}}{=} \exists f, g . \]

\[ (\forall \vec{x}) (\Phi[\vec{X}/\vec{x}] \supset (\exists i) \vec{x} = g(i)) \land \]
\[ (\forall i, j) (\Phi[\vec{X}/g(i)] \land \Phi[\vec{X}/g(j)] \land i \neq j \supset g(i) \neq g(j)) \land \]
\[ f(0) = 0 \land \]
\[ (\forall i) \left( (\Phi[\vec{X}/g(i)] \supset f(i + 1) = f(i) + \nu(g(i))) \land \right. \]
\[ \left. (\neg \Phi[\vec{X}/g(i)] \supset f(i + 1) = f(i)) \right) \land \]
\[ (\forall i) \left( f(i) \leq \nu \land \right. \]
\[ \left. (\forall v')(f(i) \leq v' \supset v \leq v') \right) \]
Robustness: \textit{Next}

\[
\begin{align*}
\text{Next}(\text{Nil}, \gamma, \delta) & \equiv \text{False} \\
\text{Next}(\alpha, \gamma, \delta) & \equiv \gamma = \alpha \land \delta = \text{Nil} \\
\text{Next}(\beta, \gamma, \delta) & \equiv \gamma = \beta \land \delta = \text{Nil} \\
\text{Next}(\phi?, \gamma, \delta) & \equiv \gamma = \phi? \land \delta = \text{Nil} \\
\text{Next}(\pi v. \sigma, \gamma, \delta) & \equiv \exists x. \text{Next}(\sigma^v_x, \gamma, \delta) \\
\text{Next}(\sigma_1 | \sigma_2, \gamma, \delta) & \equiv \text{Next}(\sigma_1, \gamma, \delta) \lor \text{Next}(\sigma_2, \gamma, \delta) \\
\text{Next}(\sigma_1; \sigma_2, \gamma, \delta) & \equiv \exists \sigma_1'. \text{Next}(\sigma_1, \gamma, \sigma_1') \land \delta = \sigma_1'; \sigma_2 \lor \\
& \text{MaybeFinal}(\sigma_1) \land \text{Next}(\sigma_2, \gamma, \delta) \\
\text{Next}(\sigma_1 || \sigma_2, \gamma, \delta) & \equiv \exists \sigma_1'. \text{Next}(\sigma_1, \gamma, \sigma_1') \land \delta = \sigma_1 || \sigma_2 \lor \\
& \exists \sigma_2'. \text{Next}(\sigma_2, \gamma, \sigma_2') \land \delta = \sigma_1 || \sigma_2' \\
\text{Next}(\sigma^*, \gamma, \delta) & \equiv \exists \sigma'. \text{Next}(\sigma, \gamma, \sigma') \land \delta = \sigma'; \sigma^*
\end{align*}
\]
Robustness: \textit{MaybeFinal}

\[
\begin{align*}
\text{MaybeFinal}(Nil) & \equiv True \\
\text{MaybeFinal}(\alpha) & \equiv False \\
\text{MaybeFinal}(\beta) & \equiv False \\
\text{MaybeFinal}(\phi?) & \equiv False \\
\text{MaybeFinal}(\pi v. \sigma) & \equiv \exists x. \text{MaybeFinal}(\sigma^v_x) \\
\text{MaybeFinal}(\sigma_1 | \sigma_2) & \equiv \text{MaybeFinal}(\sigma_1) \lor \text{MaybeFinal}(\sigma_2) \\
\text{MaybeFinal}(\sigma_1; \sigma_2) & \equiv \text{MaybeFinal}(\sigma_1) \land \text{MaybeFinal}(\sigma_2) \\
\text{MaybeFinal}(\sigma_1 \parallel \sigma_2) & \equiv \text{MaybeFinal}(\sigma_1) \land \text{MaybeFinal}(\sigma_2) \\
\text{MaybeFinal}(\sigma^*) & \equiv True
\end{align*}
\]
Robustness: $\text{transAtPr}$

\[
\text{transAtPr}(r, \alpha, \delta, s, s') = p \equiv \\
\text{if } \exists^1 \tau. \tau \geq \text{start}(s) \land \text{Poss}(\alpha[s, \tau], s) \land s' = \text{do}(\alpha[s, \tau], s) \\
\text{then } p = 1 \text{ else } p = 0
\]
Robustness: \( \text{transAtPr} \)

\[
\text{transAtPr}(r, \alpha, \delta, s, s') = p \equiv \\
\text{if } \exists \tau. \tau \geq \text{start}(s) \land \text{Poss}(\alpha[s, \tau], s) \land s' = \text{do}(\alpha[s, \tau], s) \land \\
(\forall \tau', s''. \tau' \geq \text{start}(s) \land \text{Poss}(\alpha[s, \tau'], s) \land s'' = \text{do}(\alpha[s, \tau'], s) \supset \\
\text{value}(r, \delta, s') \geq \text{value}(r, \delta, s'')) \\
\text{then } p = 1 \text{ else } p = 0
\]
Robustness: \( transAtPr \)

\[
transAtPr(r, \alpha, \delta, s, s') = p \equiv \\
\text{if } \exists^1 \tau. \tau \geq \text{start}(s) \land \text{Poss}(\alpha[s, \tau], s) \land s' = \text{do}(\alpha[s, \tau], s) \land \\
(\forall \tau', s''. \tau' \geq \text{start}(s) \land \text{Poss}(\alpha[s, \tau'], s) \land s'' = \text{do}(\alpha[s, \tau'], s) \supset \\
\text{value}(r, \delta, s') \geq \text{value}(r, \delta, s'')) \text{ then } p = 1 \text{ else } p = 0
\]

\[
transAtPr(r, \beta, \delta, s, s') = p \equiv \\
\text{if } \exists \alpha, p'. \text{Choice}(\beta, \alpha) \land \\
transAtPr(r, \alpha, \delta, s, s') \cdot \text{prob}_0(\beta, \alpha, s) = p' \land p' > 0 \\
\text{then } p = p' \text{ else } p = 0
\]
Robustness: \textit{transAtPr}

\[
\text{transAtPr}(r, \alpha, \delta, s, s') = p \equiv \\
\text{if } \exists \tau . \tau \geq \text{start}(s) \land \text{Poss}(\alpha[s, \tau], s) \land s' = \text{do}(\alpha[s, \tau], s) \land \\
(\forall \tau', s''. \tau' \geq \text{start}(s) \land \text{Poss}(\alpha[s, \tau'], s) \land s'' = \text{do}(\alpha[s, \tau'], s) \supset \\
\text{value}(r, \delta, s') \geq \text{value}(r, \delta, s'')) \text{ then } p = 1 \text{ else } p = 0
\]

\[
\text{transAtPr}(r, \beta, \delta, s, s') = p \equiv \\
\text{if } \exists \alpha, p' . \text{Choice}(\beta, \alpha) \land \\
\text{transAtPr}(r, \alpha, \delta, s, s') \cdot \text{prob}_0(\beta, \alpha, s) = p' \land p' > 0 \text{ then } p = p' \text{ else } p = 0
\]

\[
\text{transAtPr}(r, \phi?, \delta, s, s') = p \equiv \\
\text{if } \phi[s] \land s' = s \text{ then } p = 1 \text{ else } p = 0.
\]
Robustness: \( \text{transPr} \)

\[
\text{transPr}(r, \sigma, s, \delta, s') = p \equiv \\
\text{if } \exists \gamma_1, \delta_1 . \text{Next}(\sigma, \gamma_1, \delta_1) \land \\
(\forall \gamma_2, \delta_2 . \text{Next}(\sigma, \gamma_2, \delta_2) \supset \text{decomposition } \gamma_1; \delta_1 \text{ is optimal} \\
\quad \text{value}(r, (\gamma_1; \delta_1), s) \geq \text{value}(r, (\gamma_2; \delta_2), s)) \\
\text{then (if } \delta = \delta_1 \text{ then } p = \text{transAtPr}(r, \gamma_1, \delta_1, s, s') \text{ else } p = 0) \\
\text{else } p = 0
\]

execute \( \gamma_1 \)
Robustness: \( \text{transPr and Trans} \)

\[
\mathcal{D} \cup \mathcal{C} \cup \mathcal{C}' \models (\exists \delta, s') \mathcal{Trans}(\sigma, s, \delta, s') \supset
(\exists \delta, s', p) (\text{transPr}(r, \sigma, s, \delta, s') = p \land
(p > 0 \lor r(s') = 0))
\]

\[
\mathcal{D} \cup \mathcal{C} \cup \mathcal{C}' \models \text{transPr}(r, \sigma, s, \delta, s') > 0 \supset \mathcal{Trans}(\sigma, s, \delta, s')
\]
Robustness: $transPr^*$

$$transPr^*(r, \sigma, s, \delta, s') = p \overset{\text{def}}{=}$$

\[ \text{if } \exists p'. \forall f. (\forall r', \sigma_1, s_0. f(r', \sigma_1, s_0, \sigma_1, s_0) = 1) \land \]
\[ (\forall r', \sigma_1, \delta_1, \delta_2, s_0, s_1, s_2, p_1, p_2. \]
\[ p_1 > 0 \land f(r', \sigma_1, s_0, \delta_1, s_1) = p_1 \land \]
\[ p_2 > 0 \land transPr(r', \delta_1, s_1, \delta_2, s_2) = p_2 \supset \]
\[ f(r', \sigma_1, s_0, \delta_2, s_2) = p_1 \cdot p_2) \supset \]
\[ f(r, \sigma, s, \delta, s') = p' \]

\[ \text{then } p = p' \text{ else } p = 0 \]
Robustness: $Final$

\[
Final(r, \sigma, s) \equiv MaybeFinal(\sigma) \land \\
value(r, Nil, s) \geq value(r, \sigma, s)
\]
Robustness: \( doPr^* \)

\[
doPr(r, \sigma, s, s') = p \overset{\text{def}}{=} \\
\text{if } \exists p'. \; \text{transPr}^*(r, \sigma, s, s') = p' \land \text{Final}(r, \sigma, s') \land \\
(\forall s'')(s \sqsubseteq s'' \land s'' \sqsubseteq s' \supset \neg \text{Final}(r, \sigma, s'')) \\
\text{then } p = p' \text{ else } p = 0
\]
Atomic Complex Actions: Semantics

\[ \text{Next}(\text{atomic}(\sigma), \gamma, \delta) \equiv \gamma = \text{atomic}(\sigma) \land \delta = \text{Nil}. \]

\[ \text{Next}'(\sigma, \gamma, \delta) \overset{\text{def}}{=} \forall P. (\forall \sigma', \gamma', \delta'. \text{Next}(\sigma', \gamma', \delta') \supset P(\sigma', \gamma', \delta')) \land \\
(\forall \sigma', \sigma'', \gamma', \gamma'', \delta', \delta''. \ P(\sigma', \gamma', \delta') \land \gamma' = \text{atomic}(\sigma'') \land \\
\text{Next}(\sigma''; \delta', \gamma'', \delta'') \supset \\
P(\sigma', \gamma'', \delta'')) \supset \\
P(\sigma, \gamma, \delta) \land (\forall \sigma') \gamma \neq \text{atomic}(\sigma') \]
Atomic Complex Actions: Plan Recognition

Candidate program:

- Make db inconsistent at $\tau_2 = 2$
- Regain consistency at $\tau_3 = 2$

Observations:

- $\tau_1 = 1$: $\phi_1 = \text{"db cons."}$
- $\tau_2 = 2$: $\phi_2 = \text{"db incons."}$
- $\tau_3 = 2$: $\phi_3 = \text{"db cons."}$

Is $(\tau_2, \phi_2)$ observable? No!

Inconsistent situation has timespan zero
Atomic Complex Actions: Plan Recognition

Candidate program:

▶ Make db inconsistent at \( \tau_2 = 2 \)
▶ Regain consistency at \( \tau_3 = 2 \)

Observations:

- \( \tau_1 = 1 \): \( \phi_1 = \text{"db cons."} \)
- \( \tau_2 = 2 \): \( \phi_2 = \text{"db incons."} \)
- \( \tau_3 = 2 \): \( \phi_3 = \text{"db cons."} \)

Should \( observe(\tau_2, \phi_2) \) be executable? No! But it is!

\[
\sigma \parallel (\ldots; \text{atomic}(observe(\tau_2, \phi_2); waitFor(now > \tau_2)); \ldots)
\]