Towards Reasoning About Teleo-Reactive Programs

Ian J. Hayes

School of ITEE
The University of Queensland
Brisbane, 4072, Australia

School of Computing Science
University of Newcastle
Newcastle, UK
Until Jan 2009

Email: ianh@itee.uq.edu.au

Serene Presentation 19 November 2008

Resilience: an audience that turns up the morning after the conference dinner

Overview

Teleo-reactive programming was invented by Nils Nilsson [6, 7, 8]. We

- extend teleo-reactive programs with non-determinism
- develop a time-interval semantics
- develop rely/guarantee reasoning rules and
- apply the rules to an example program

(Greek: telos: end, purpose)

Teleo-reactive Programs

Teleo-reactive programs are described by a combination of

- sensed values (inputs), or values derived or inferred from sensed values,
- primitive durative actions, i.e., actions that take time, and
- processes defined via (durative) prioritised conditionals.

Nilsson's Tower Program

makeTower(s) — s is an non-empty list with no duplicates

$$
\begin{array}{rcl}
Tower(s) & \rightarrow & nil, \\
Ordered(s) & \rightarrow & unpile(head(s)), \\
Null(tail(s)) & \rightarrow & move\_to\_table(head(s)), \\
Tower(tail(s)) & \rightarrow & move(head(s), head(tail(s))), \\
true & \rightarrow & makeTower(tail(s))
\end{array}
$$

move_to_table(x) — x is a block

$$
\begin{array}{rcl}
On(x, Ta) & \rightarrow & nil, \\
Holding(x) & \rightarrow & put\_down(x, Ta), \\
Clear(x) & \rightarrow & pick\_up(x), \\
true & \rightarrow & unpile(x)
\end{array}
$$

move(x,y) — x and y are distinct blocks

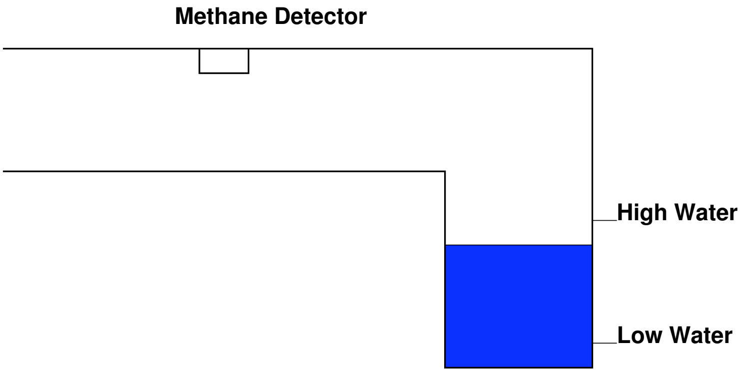
$$
\begin{array}{rcl}
On(x,y) & \rightarrow & nil, \\
Holding(x) \wedge Clear(y) & \rightarrow & put\_down(x,y), \\
Holding(z) \wedge x \neq z & \rightarrow & put\_down(z,Ta), \\
Clear(x) \wedge Clear(y) & \rightarrow & pick\_up(x), \\
Clear(y) & \rightarrow & unpile(x), \\
true & \rightarrow & unpile(y)
\end{array}
$$

unpile(x) — x is a block

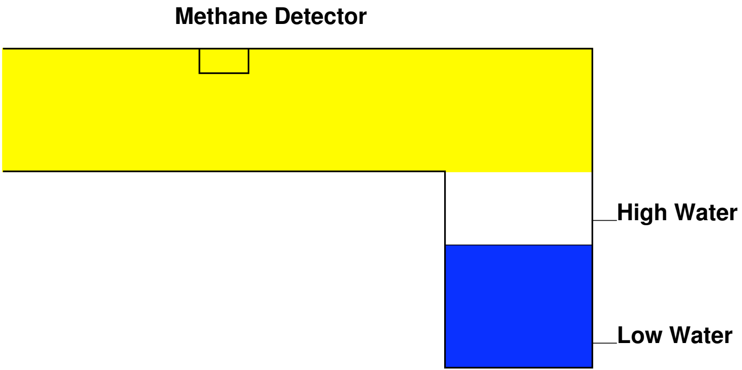
$$
\begin{array}{rcl}
Clear(x) & \rightarrow & nil, \\
On(y,x) & \rightarrow & move\_to\_table(y)
\end{array}
$$

pick_up and *put_down* are primitive actions.

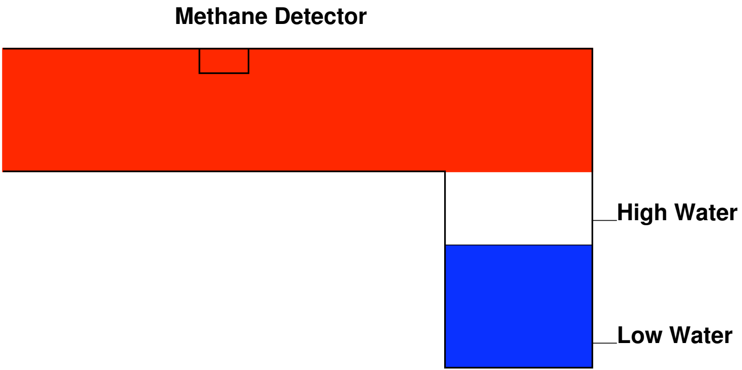
Mine with sump



Mine with methane



Mine with methane - explodes if pump on



Mine Pump Example [3, 5, 4]

mine_pump

$$\begin{array}{lcl} \mathit{Critical} \leq \mathit{methane} & \rightarrow & \text{alarm}, \\ \mathit{true} & \rightarrow & \text{operate} \end{array}$$

operate

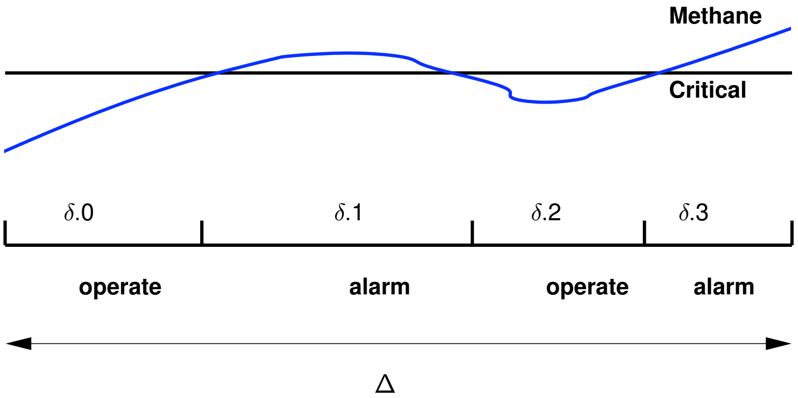
$$\begin{array}{lcl} \left(\begin{array}{l} \mathit{water} > \mathit{High} \vee \\ \mathit{water} > \mathit{Low} \wedge \mathit{pump\_active} \end{array} \right) & \rightarrow & \text{pump}, \\ \mathit{true} & \rightarrow & \text{nil} \end{array}$$

There are three sensed values:

- the level of methane in the mine, *methane*,
- the level of water in the mine, *water*, and
- an indicator of whether the pump is active, *pump_active*.

It is not an Action System

Durative actions



Expanded Program

The above is equivalent to the following,

mine_pump

$$
\begin{array}{lcl}
\mathit{Critical} \leq \mathit{methane} & \rightarrow & \text{alarm}, \\
\mathit{true} \wedge \left(\begin{array}{l} \mathit{water} > \mathit{High} \vee \\ \mathit{water} > \mathit{Low} \wedge \mathit{pump\_active} \end{array} \right) & \rightarrow & \text{pump}, \\
\mathit{true} \wedge \mathit{true} & \rightarrow & \text{nil}
\end{array}
$$

From this it is easy to see that the pump is only ever active while the methane level is not critical and the water level is at least above low.

Rely/Guarantee for the Pump

When the primitive action pump is active it guarantees

Pump guarantee

$$g\_pump \mathrel{\widehat{=}} \Box(MinOut \leq water\_out \wedge pump\_active)$$

but relies on

Pump rely

$$r\_pump \mathrel{\widehat{=}} \Box(Low < water \wedge methane < Critical)$$

Nondeterminism choice for deterministic procedure

For example, the mine_pump procedure

mine_pump

$$\begin{array}{rcl} Critical \leq methane & \rightarrow & \text{alarm} \\ \sqcap\ methane < Critical & \rightarrow & \text{operate} \end{array}$$

is equivalent to earlier:

mine_pump

$$\begin{array}{lcl} Critical \leq methane & \rightarrow & \text{alarm}, \\ true & \rightarrow & \text{operate} \end{array}$$

Nondeterminism

In general, the guard conditions of a nondeterministic choice do not need to be mutually exclusive. The following

operate specification

$$\begin{array}{rcl} water > Low & \rightarrow & \text{pump} \\ \sqcap\ water \leq High & \rightarrow & \text{nil} \end{array}$$

is refined by earlier:

operate

$$\begin{array}{lcl} \left(\begin{array}{l} water > High \vee \\ water > Low \wedge pump\_active \end{array} \right) & \rightarrow & \text{pump}, \\ true & \rightarrow & \text{nil} \end{array}$$

Conditional

A conditional

$$c0 \rightarrow a0,$$
$$c1 \rightarrow a1$$

is expanded to

$$\begin{array}{rrl} & c0 & \rightarrow a0 \\ \sqcap & \neg c0 \wedge c1 & \rightarrow a1 \\ \sqcap & \neg c0 \wedge \neg c1 & \rightarrow chaos \end{array}$$

Timed traces and intervals

The semantics of a teleo-reactive program are given by specifying its set of behaviours over a given time interval.
A behaviour is a trace of the values of program's variables over time.
If c is a predicate (condition) on the state of the program variables, we use the notation $\Box c$ as a predicate over a time interval, Δ, that states that condition c holds at all times in Δ, i.e., $(\Box c).\Delta$.

Basic Behaviours

For a predicate c; actions a, a_0, a_1; and a time interval Δ

bbeh

$$bbeh.chaos.\Delta \;\widehat{=}\; True \tag{1}$$

$$bbeh.(c \rightarrow a).\Delta \;\widehat{=}\; (\Box c).\Delta \wedge beh.a.\Delta \tag{2}$$

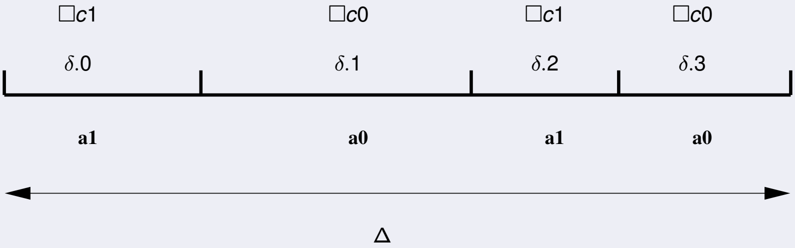
$$bbeh.(a_0 \sqcap a_1).\Delta \;\widehat{=}\; bbeh.a_0.\Delta \vee bbeh.a_1.\Delta \tag{3}$$

Behaviours

beh

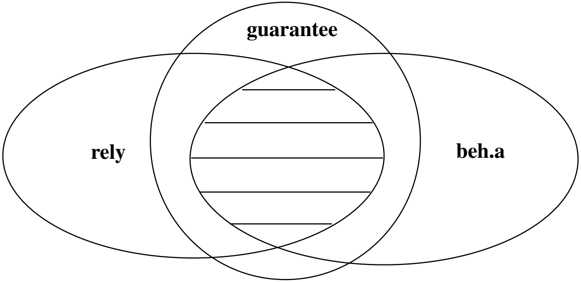
$$beh.a.\Delta \ \widehat{=}\ \exists \delta : part.\Delta \bullet \forall i : \mathrm{dom}\,.\delta \bullet bbeh.a.(\delta.i) \qquad (4)$$

A partition for "$c0 \rightarrow a0, c1 \rightarrow a1$"



We use the notation $part.\Delta$ for the set of all partitions of an interval Δ.

Satisfying a Rely/Guarantee Pair



Definition (Satisfies)

An action, *a*, *satisfies* a rely/guarantee pair, written "$r\ \{a\}\ g$", if all behaviours of *a* for which the rely condition holds also satisfy the guarantee condition, i.e.,

$$beh.a \wedge r \Rrightarrow g$$

Basic Rely/Guarantee Theorem

Theorem (Basic rely/guarantee)

For rely conditions r and r', guarantee conditions g and g', and an action a, if

$$r \Rrightarrow r'$$
$$r' \; \{a\} \; g'$$
$$g' \land r \Rrightarrow g$$

then

$$r \; \{a\} \; g$$

Guarded Actions

Theorem (Guarded action)

For a rely condition r, a guarantee condition g, a predicate c, and an action a,

$$r\ \{c \rightarrow a\}\ g \equiv (\Box c \wedge r)\ \{a\}\ g$$

Decomposing (Rely) Conditions

Definition (Decomposes)

A (rely) condition, r, *decomposes over intervals* if whenever r holds for an interval Δ, it holds for its subintervals:

$$\forall \Delta, \Delta' : \mathit{Interval} \bullet \Delta' \subseteq \Delta \land r.\Delta \Rrightarrow r.\Delta'$$

For example, $\Box c$ decomposes over intervals,
but $r.\Delta \mathrel{\widehat{=}} \mathit{length}.\Delta \geq 10$ does not.

Composing (Guarantee) Conditions

Definition (Composes)

A (guarantee) condition, *g*, *composes over intervals* if whenever *g* holds for every subinterval in a partition δ of Δ, it holds for Δ:

$$\forall \Delta : \mathit{Interval} \bullet \forall \delta : \mathit{part}.\Delta \bullet$$
$$(\forall i : \mathrm{dom}\,.\delta \bullet g.(\delta.i)) \Rrightarrow g.\Delta$$

For example, $\Box c$ composes over intervals,
and hence $\Box(x = 0 \vee x = 1)$ composes
but $\Box(x = 0) \vee \Box(x = 1)$ does not.

Nondeterminism

Theorem (Nondeterminism)

For rely condition r that decomposes over intervals, guarantee condition g that composes over intervals, and actions a_0 and a_1,

$$r\ \{a_0 \sqcap a_1\}\ g$$

provided

$$(r\ \{a_0\}\ g) \wedge (r\ \{a_1\}\ g)$$

Two-Branch Conditional

Theorem (Conditional)

For a rely condition, r, that decomposes over intervals, a guarantee condition, g, that composes over intervals, predicates, c_0 and c_1, actions, a_0 and a_1,

$$r\ \{c_0 \rightarrow a_0,\, c_1 \rightarrow a_1\}\ g$$

provided

$$((\Box c_0) \wedge r)\ \{a_0\}\ g \qquad (5)$$

$$(\Box(\neg c_0 \wedge c_1) \wedge r)\ \{a_1\}\ g \qquad (6)$$

$$r \Rrightarrow \Box(c_0 \vee c_1) \qquad (7)$$

The above theorem can be generalised.

Example: Mine Pump

water	the level of water in the mine shaft
$\frac{dwater}{dt}$	the derivative of the water level with respect to time, i.e., the rate of change of the water level.
water_in	the rate of flow of water into the mine
water_out	the rate of flow of water out of the mine

Rely condition for mine pump system

$$r \mathrel{\widehat{=}} \Box \left(\begin{array}{l} \frac{dwater}{dt} = water\_in - water\_out \wedge \\ 0 \leq water\_out \wedge \\ 0 \leq water\_in \leq MaxIn \end{array} \right)$$

Pump

When the primitive action pump is active it guarantees

Guarantee for pump

$$g\_pump \mathrel{\widehat{=}} \Box(MinOut \leq water\_out \wedge pump\_active)$$

but relies on

Rely for pump

$$r\_pump \mathrel{\widehat{=}} \Box(Low < water \wedge methane < Critical)$$

Guarantee

We would like to show our mine pump system guarantees

Guarantee for the mine pump system

$$g \mathrel{\widehat{=}} \Box \left(\begin{array}{l} \mathit{methane} < \mathit{Critical} \wedge \mathit{High} < \mathit{water} \\ \Longrightarrow \\ \frac{\mathit{dwater}}{\mathit{dt}} \leq \mathit{MaxIn} - \mathit{MinOut} \end{array} \right)$$

which, provided that $\mathit{MaxIn} < \mathit{MinOut}$, guarantees that the water level will decrease.

Mine Pump Rely/Guarantee

mine_pump

$$\begin{array}{lcl} \mathit{Critical} \leq \mathit{methane} & \rightarrow & \text{alarm}, \\ \mathit{true} & \rightarrow & \text{operate} \end{array}$$

satisfies the rely/guarantee pair, i.e.,

$$r \; \{\text{mine\_pump}\} \; g$$

provided

$$(\Box(\mathit{Critical} \leq \mathit{methane}) \wedge r) \; \{\text{alarm}\} \; g$$
$$(\Box(\mathit{methane} < \mathit{Critical}) \wedge r) \; \{\text{operate}\} \; g$$
$$r \Rrightarrow \Box(\mathit{Critical} \leq \mathit{methane} \vee \mathit{true})$$

$$(\Box(\mathit{Critical} \leq \mathit{methane}) \wedge r)\ \{\text{alarm}\}\ g \tag{8}$$

$$(\Box(\mathit{methane} < \mathit{Critical}) \wedge r)\ \{\text{operate}\}\ g \tag{9}$$

$$r \Rrightarrow \Box(\mathit{Critical} \leq \mathit{methane} \vee \mathit{true}) \tag{10}$$

Requirement (10) holds trivially.
Requirement (8) holds because $\mathit{Critical} \leq \mathit{methane}$ is the complement of $\mathit{methane} < \mathit{Critical}$ used on the left side of the implication within g.

For requirement (9) we need to show procedure operate

operate

$$\begin{array}{lcl} \left(\begin{array}{l} water > High \vee \\ water > Low \wedge pump\_active \end{array}\right) & \rightarrow & \text{pump}, \\ true & \rightarrow & \text{nil} \end{array}$$

satisfies

$$(\Box(methane < Critical) \wedge r)\ \{\text{operate}\}\ g$$

Let c be the guard on the first branch of procedure operate. The conditions we need to show are

$$\begin{array}{l} (\Box(c \wedge methane < Critical) \wedge r)\ \{\text{pump}\}\ g \\ (\Box(\neg c \wedge methane < Critical) \wedge r)\ \{\text{nil}\}\ g \\ \Box(methane < Critical) \wedge r \Rrightarrow \Box(c \vee true) \end{array}$$

$$(\Box(c \wedge methane < Critical) \wedge r)\ \{\text{pump}\}\ g \tag{11}$$

$$(\Box(\neg c \wedge methane < Critical) \wedge r)\ \{\text{nil}\}\ g \tag{12}$$

$$\Box(methane < Critical) \wedge r \Rrightarrow \Box(c \vee true) \tag{13}$$

Requirement (13) is trivial.
For (12), the negation of c implies that $water \leq High$, which implies the left side of the implication in g is false.
Requirement (11) follows from the rely and guarantee conditions of pump, provided

$$\begin{aligned} \Box(c \wedge methane < Critical) \wedge r &\Rrightarrow r\_pump \\ g\_pump \wedge \Box(c \wedge methane < Critical) \wedge r &\Rrightarrow g \end{aligned}$$

$$\begin{array}{rcl} \Box(c \wedge methane < Critical) \wedge r & \Rrightarrow & r\_pump \\ g\_pump \wedge \Box(c \wedge methane < Critical) \wedge r & \Rrightarrow & g \end{array}$$

The first requirement holds because c implies $water > Low$. For the second requirement, from r we have

$$\begin{array}{l} \frac{dwater}{dt} = water\_in - water\_out \\ \Rrightarrow \text{as } g\_pump \text{ implies } MinOut \leq water\_out \\ \frac{dwater}{dt} \leq water\_in - MinOut \\ \Rrightarrow \text{as } r \text{ implies } water\_in \leq MaxIn \\ \frac{dwater}{dt} \leq MaxIn - MinOut \end{array}$$

Conclusions

The teleo-reactive programming paradigm of Nilsson provides a remarkably simple notation for expressing robust real-time control programs. In this paper we have

- developed a time-interval semantics for teleo-reactive programs and
- provided rely/guarantee reasoning rules that have been shown correct with respect to these rules.

Future Work

For some teleo-reactive programs guarded actions may only be active for an instant during which time they change the state to disable themselves. There are two approaches we can follow to address these instantaneous actions, either

- enrich our semantics to allow a finite number of instantaneous actions to happen at the same real time, or
- make use of the notion of time bands [2, 1].

For the latter, a system description can be structured into a number of time bands each with its timing precision. At a higher-level time band an action may be considered instantaneous if it is within the precision of the band, but within a lower level band the same action may be viewed as taking time.

References

G. Baxter, A. Burns, and K. Tan.
Evaluating timebands as a tool for structuring the design of socio-technical systems.
In P. Bust, editor, *Contemporary Ergonomics 2007*, pages 55–60. Taylor and Francis, 2007.

A. Burns and G. Baxter.
Time bands in systems structure, pages 74–88.
Springer-Verlag, 2006.

A. Burns and A. Lister.
A framework for building dependable systems.
Computer Journal, 34(2):173–181, 1991.

M. Joseph, editor.
Real-time Systems: Specification, Verification and Analysis.
Prentice Hall, 1996.

B. P. Mahony and I. J. Hayes.
A case-study in timed refinement: A mine pump.
IEEE Trans. on Software Engineering, 18(9):817–826, 1992.

N. Nilsson.
Teleo-reactive programs for agent control.
Journal of Artificial Intelligence Research, 1:139–158, 1994.

N. J. Nilsson.
Teleo-reactive programs and the triple-tower architecture.
Electronic Transactions on Artificial Intelligence, 5:99–110, 2001.

N. Nilsson.
Teleo-reactive programming web site, Last accessed 2008.
http://robotics.stanford.edu/users/nilsson/trweb/tr.html.

Acknowledgments

This paper has benefited from my collaborations with members of IFIP Working Group 2.3 on Programming Methodology as well as Cliff Jones, Alan Burns, and Keith Clark (who introduced me to teleo-reactive programming). This research was supported by Australian Research Council (ARC) Discovery Grant DP0558408, *Analysing and generating fault-tolerant real-time systems* and the EPSRC-funded Trustworthy Ambient Systems (TrAmS) Platform Project.