AXIOMATIC FORMAL METHODS IN REAL-TIME SYSTEM DESIGN

Axiomatic proof systems - aim is to improve the conciseness and precision of software design.

For general purpose software there are several formal specification languages, e.g. Z and VDM. These proof systems do not deal explicitly with timing properties -> difficult to use for real-time performance verification.

There are significant limitations on the axiomatic method, even without considering time constraints:

- Time dependent properties in concurrent systems are difficult to specify (e.g. mutual exclusion & concurrency)
- Determining invariants in complex systems is difficult
- Complex expression simplification is tedious (and error-prone)

Four axiomatic approaches can be mentioned:

- **Dijkstra's weakest precondition** - definitions can be used to derive a latest-time to begin a computation
- **Real-Time Logic** - can be used to specify and infer temporal properties
- **Time-related history variables** - can be used to reason about temporal properties
- **Extended state machines and Real-Time Temporal Logic** - combined automata model and axiomatic temporal constraint specification

REAL-TIME LOGIC (RTL)

First proposed by Jahanian & Mok [1986], the RTL computational model is made up of:

- events
- actions
- causality relations
- timing constraints

and formally constructed with first-order predicate logic.

An *event occurrence function* is defined for an event *e* at the i^{th} occurrence as (*e*, *i*) and the time of occurrence is @(*e*, *i*).

There are three types of constant in RTL:

- *action constants* - primitive or composite, where a composite constant has a precedence imposed by the causality relations between actions.
- *event constants* - 3 subtypes:
  - *start/stop events* for action initiation/termination
  - *transition events* for changes in state behaviour
  - *external events* which affect system behaviour but are not system caused
- *integers* - time values or event numbers

RTL axioms are derived from the event-action model of the system by characterizing:

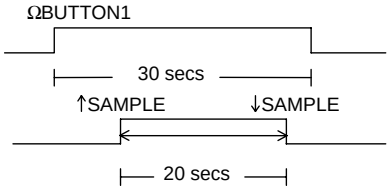
- Relations between actions and their start/stop events
- Sporadic and periodic event constraints
- Causal relations which may generate a transition event
- Additional constraints to prevent action occurrence

RTL Example 1

Consider the following specification:

A panel with a button that when pushed must generate an action SAMPLE which must execute within 30 seconds. The computation time of this SAMPLE action is at least 20 seconds.

We can illustrate this with a timing diagram (as shown below) and specify this example with three RTL axioms:



$$\forall x\text{: } @(\Omega\text{BUTTON1}, x) \leq @(\uparrow\text{SAMPLE}, x)$$
$$\wedge\ @(\downarrow\text{SAMPLE}, x) \leq @(\Omega\text{BUTTON1}, x) + 30$$
$$\forall y\text{: } @(\uparrow\text{SAMPLE}, y) + 20 \leq @(\downarrow\text{SAMPLE}, y)$$

where x and y are integer event counters, and the axioms are true for all values of x and y.

Interpretation:

- 1st axiom - the time of BUTTON1 depression must be less than or equal to the start time of action SAMPLE.
- 2nd axiom - the end time of action SAMPLE must be less than or equal to the time of BUTTON1 depression plus 30 seconds.
- 3rd axiom - the end time of action SAMPLE must be equal to or exceed the start time of action SAMPLE by 20 seconds.

RTL Constraint Graphs

Each RTL axiom can be reduced to a constraint and a transformed occurrence function.

The transformed RTL axioms are constructs in *Presburger arithmetic* which consists of inequality predicates, the addition and subtraction operator, first-order logic connectives and uninterpreted integer functions. Each occurrence function $@(e, i)$ is transformed to an *uninterpreted function* $f_e(i)$. For the above example, the uninterpreted function formulas are:

$$\forall x\text{: } f_1(x) \leq f_2(x) \wedge f_3(x) \leq f_1(x) + 30$$
$$\forall y\text{: } f_2(y) + 20 \leq f_3(y)$$

The conversion of RTL axioms to uninterpreted function expressions follows the same procedure as the conversion of Well-formed formulas (Wff) in Predicate Calculus to Clausal form.

The set of transformed RTL axioms (F) converted to Presburger arithmetic formulas (F') can be put in conjunctive normal form CNF (F''). The general form of a formula in CNF is:

$$F'' = C_1 \wedge C_2 \wedge \cdots \wedge C_n$$

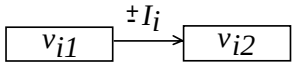
where each C_i is a disjunctive clause of the form:

$$C_i = L_1 \vee L_2 \vee \cdots \vee L_m$$

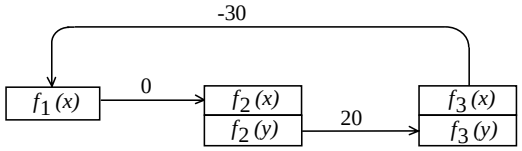
and each L_j is a literal of the form:

$$L_i\text{: } v_{i1} \pm I_i \leq v_{i2}$$

where each v_{i1} and v_{i2} are uninterpreted integer functions and I_i is an integer constant. Each literal is represented as two nodes joined by an arc on the RTL constraint graph:

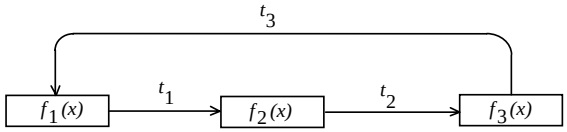


For the example, the following RTL constraint graph can be drawn:



RTL constraint graph node reduction

Take the above example with the variable substitution, $\Psi = \{y \rightarrow x\}$ which by inspection allows the reduced graph below to be constructed. Just for illustration purposes we generalize the time constraints also (to t_1, t_2 & t_3):



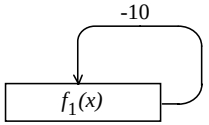
The application of node reduction techniques to each arc of the graph is mapped to the deductive resolution of the corresponding RTL axioms, e.g:

$$\{f_1(x) + t_1 \leq f_2(x)\} \wedge \{f_2(x) + t_2 \leq f_3(x)\} \wedge \{f_3(x) + t_3 \leq f_1(x)\}$$
$$\Rightarrow f_1(x) + t_1 + t_2 + t_3 \leq f_1(x)$$
$$\Rightarrow t_1 + t_2 + t_3 \leq 0$$

i.e. the sum of these time constraints must be less than or equal to zero otherwise the inequality fails → RTL axioms are not *consistent* → timing specification cannot be met.

Thus, any *positive* cycle in the reduced RTL constraint graph implies that the timing specification for the system cannot be met.

Returning to the example, after node reduction we have a single negative cycle which implies that the timing specification can be met:



If we change the specification:
Action SAMPLE must now be executed within 15 time units of the button being pressed
→ a positive cycle which implies that the specification cannot be now be met.

Testing a timing property against the system specification

This can be generalized by applying any timing property or safety assertion (which is just a set of RTL axioms $\{P\}$) and check it for consistency against the system specification $\{S\}$.

We can conjoin the negation of the timing property $\{P\}$ with the system specification $\{S\}$:

$$\neg\{P\} \wedge \{S\} \rightarrow \text{unsatisfiable}$$

which must be false or unsatisfiable (i.e. have at least one *positive* cycle) for the timing property to be consistent with the system specification. In a purely analogous form, graphically we have:

unsatisfiable -> consistent **satisfiable -> inconsistent**

i.e. if the timing property $\{P\}$ contains (and hence does not contradict any axiom in) the system specification $\{S\}$, then there is no occurrence function created, i.e. it is unsatisfiable. $\rightarrow$ the timing property is consistent with the system specification. But if the timing property $\{P\}$ effectively removes or invalidates any axiom from the system spec $\{S\}$, $\rightarrow$ some occurrence function is created, i.e. it is not consistent with the system specification.

RTL Example 2

A safety assertion is added to Example 1:

If the transmitted information is displayed within 10 time units of the completion of action SAMPLE, then within 40 time units of pressing button 1, the requested information will be displayed.

In RTL notation the safety assertion or property $\{P\}$ is:

$$\forall u \, \forall t\text{: } @\,(\downarrow \text{SAMPLE}, u) \;\le\; @\,(\Omega\,\text{DISPLAY}, t)$$
$$\wedge\; @\,(\Omega\,\text{DISPLAY}, t) \;\le\; @\,(\downarrow \text{SAMPLE}, u) + 10$$
$$\rightarrow\; @\,(\Omega\,\text{BUTTON1}, u) \;<\; @\,(\Omega\,\text{DISPLAY}, t)$$
$$\wedge\; @\,(\Omega\,\text{DISPLAY}, t) \;\le\; @\,(\Omega\,\text{BUTTON1}, u) + 40$$

The negated form of the safety assertion ($\neg\{P\}$) is given by:

$$\exists u \exists t\text{: } @\,(\downarrow \text{SAMPLE}, u) \;\le\; @\,(\Omega\,\text{DISPLAY}, t)$$
$$\wedge\; @\,(\Omega\,\text{DISPLAY}, t) \;\le\; @\,(\downarrow \text{SAMPLE}, u) + 10$$
$$\wedge\; \{\, @\,(\Omega\,\text{DISPLAY}, t) \;\le\; @\,(\Omega\,\text{BUTTON1}, u)$$
$$\vee\; @\,(\Omega\,\text{BUTTON1}, u) + 41 \le @\,(\Omega\,\text{DISPLAY}, t)\}$$

We note that since the time constants are integers, in the negation of the last inequality above, the following relation is being utilized:

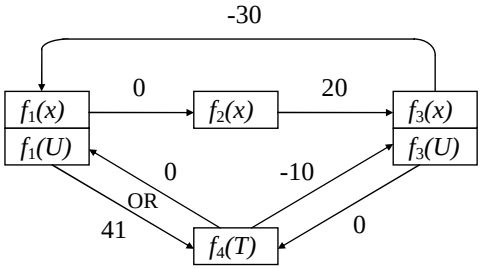
$$\neg\{@(E_1, i) \le @(E_2, j)\} = @(E_2, j) + 1 \le @(E_1, i)$$

We can also observe that the universal quantifies are replaced by existential quantifiers in the negated form and the logical conjunction is converted to a disjunction. The RTL axioms are transformed to formulas in Presburger arithmetic with uninterpreted integer functions:

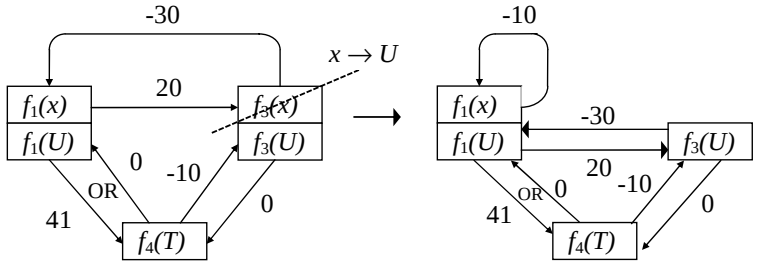
$$f_3(U) \;\le\; f_4(T) \wedge f_4(T) \;\le\; f_3(U) + 10$$
$$f_4(T) \;\le\; f_1(U) \vee f_1(U) + 41 \;\le\; f_4(T)$$

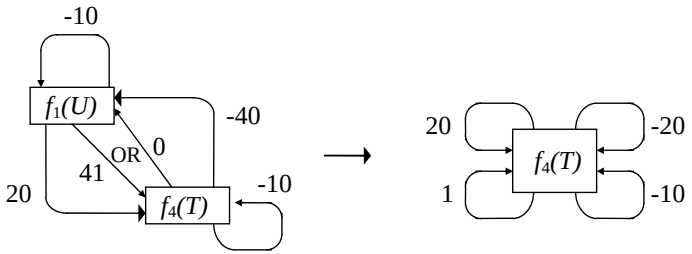
The existentially quantified variables *u* and *t* are converted to *Skolem constants* *U* and *T* (i.e. instances of the variables that satisfy the quantifier).

The RTL constraint graph can then be produced for this example:



Nodes are eliminated one-by-one, by making substitutions for variables. All cycles that can be generated through eliminated nodes must be included on the reduced graph, e.g:





Note that disjunctive literals must appear in different cycles to influence the satisfiability and we must have at least one positive cycle in all disjunctive possibilities to make the conjoined system satisfiable.

Thus in this example the safety assertion produces an unsatisfiable $\neg \{P\} \wedge \{S\}$ -> the safety assertion is consistent with the system specification.