Challenges and Promises in Reactive Modeling: A Personal Perspective

Tom Henzinger
What happens if you push that button?
Formal Methods for Complex Reactive Systems

1. Model design
2. Model analysis
Some Paradigmatic Modeling Languages

Moore/Mealy machines
Petri nets
Kahn networks
Threads
CSP/CCS
Synchronous languages
Statecharts
Timed automata
Pi-calculus
etc.
Some Personal Modeling Languages

**Hybrid automata**: mixing discrete and continuous behavior  
**Reactive modules**: mixing synchronous and asynchronous interaction  
**Interface automata**: independent implementability of components  
**Giotto**: time-triggered task coordination
Some Personal Modeling Languages

**Hybrid automata**: mixing discrete and continuous behavior
**Reactive modules**: mixing synchronous and asynchronous interaction
**Interface automata**: independent implementability of components
**Giotto**: time-triggered task coordination

Some Personal Specification Languages

**Game temporal logics**: specifying cooperation and competition
**Nested weighted automata**: specifying resource consumption
**Pre/post fixpoint calculus**: specifying symbolic state-space exploration
More and Less Solved Issues in Language Design

Concurrency
Mobility
Modularity
Strategic aspects
Quantitative aspects
More and Less Solved
Modularity Aspects in Language Design

Composition and communication
Spatial abstraction

Temporal abstraction
Shared refinement
Mixing data flow and control flow
Graphical System Models

Engineering
Component model: transfer equation
Composition: spatial (parallel)
Connection: data flow

Computer Science
Component model: subroutine
Composition: temporal (sequential)
Connection: control flow
Abstraction Hierarchies

Spatial

Temporal
Top-down Design

Operations:
1. Parallel composition $\parallel$
2. Refinement $\%$
Independent Implementability
If $A$ and $B$ are composable and $A' \leq A$ and $B' \leq B$, then $A'$ and $B'$ are composable and $A'||B' \leq A||B$. 
Component-based Design: Shared Refinement
Multiple Viewpoints: Shared Refinement

- Functional view $F$
- Timing view $T$
- Power view $P$

Graphical representation shown in the image.
Stateless Interface

Well-formedness:
1. Assumption satisfiable
2. Guarantee satisfiable
Parallel Composition

\[ \text{even}(x) \rightarrow A \rightarrow y \mod 3 = 0 \]

\[ x > 0 \rightarrow B \rightarrow z \mod 4 = 0 \]

\[ A || B \]

\[ \text{even}(x) \rightarrow A \rightarrow y \]

\[ x > 0 \rightarrow B \rightarrow z \]

\[ y \mod 3 = 0 \]

\[ z \mod 4 = 0 \]
Parallel Composition

\[\text{even}(x) \rightarrow A \rightarrow y \mod 3 = 0\]

\[\text{odd}(x) \rightarrow B \rightarrow z \mod 4 = 0\]

\[\text{false} \rightarrow x \rightarrow \neg\]

\[\neg \rightarrow A \rightarrow y \mod 3 = 0\]

\[\neg \rightarrow B \rightarrow z \mod 4 = 0\]

\[A || B\]
Refinement

B refines A

even(x) even(y)

x int y mod 4 = 0
B does not refine C:
Implementation may produce inadmissible outputs.
Refinement

B does not refine D:
Implementation does not accept all admissible inputs.
Shared Refinement

\[ \text{even}(x) \rightarrow A \rightarrow y \mod 3 = 0 \]

\[ \text{x} > 0 \rightarrow B \rightarrow y \mod 4 = 0 \]

\[ \forall x \in \mathbb{N}, x > 0 \rightarrow A \Pi B \rightarrow y \mod 12 = 0 \]

\[ \text{even}(x) \rightarrow A \Pi B \]

A \Pi B can be used in any design as an implementation of A, and as an implementation of B.
Shared Refinement

\[ \text{even}(x) \]

\[ \text{odd}(y) \]

\[ \text{x>0} \]

\[ \text{y mod 4 = 0} \]

\[ \text{A \land B} \rightarrow \text{false} \]

\[ \text{NOT SHARED-REFINABLE!} \]
Modularity Challenge:

Define a modeling language with feedback, independent implementability, and shared refinement.
More and Less Solved
Strategic Aspects in Language Design

Controller synthesis

Competitive/cooperative composition
Strategy-preserving abstraction
Behavioral equilibria
Stateless Interface

This interface constrains the client’s data.
Stateful Interface

This interface constrains the client’s control.
Stateful Interface Compatibility

Client1
- open!
- close!
- read!
- data?
- data

File
- open?
- close?
- open?
- close?
- read?
- read?
- data!
- data
Stateful Interface Incompatibility

Client2

open!
open!
open!

File

open?
close?
read?
data!

open
close
read
data
Stateful Interface Composition
Stateful Interface Composition
Composition as Game:
Composite Interface
Two Threads

$P_1$:
init $x := 0$
loop
  choice
    | $x := x + 1 \mod 2$
    | $x := 0$
  end choice
end loop

$\Phi_1: \Box (x \neq y)$

$P_2$:
init $y := 0$
loop
  choice
    | $y := x$
    | $y := x + 1 \mod 2$
  end choice
end loop

$\Phi_2: \Box \text{even}(y)$
Graph Questions

\[ x \quad \square (x \preceq y) \]
\[ \checkmark \quad \triangle (x \preceq y) \]
Game Questions

\[
\begin{align*}
X & \quad ; \#\Box (x \not< y) \\
\checkmark & \quad <\#\Box (x \not< y)
\end{align*}
\]

\[
\begin{align*}
& \quad \lceil \kappa \mathcal{P}_1 \rceil \ll \Box (x \not< y) \\
& \quad \lceil \kappa \mathcal{P}_2 \rceil \ll \Box \text{even}(y)
\end{align*}
\]
Game Questions

\[ X \quad ; \quad \# (x \not\approx y) \]

\[ \checkmark \quad <\# (x \not\approx y) \]

\[ X \quad \mathbb{G} P_1 \mathbb{L} \quad (x \not\approx y) \]

\[ \checkmark \quad \mathbb{G} P_2 \mathbb{L} \quad \text{even}(y) \]
Equilibrium Question

$\mathbb{P}_1 \cup (x \neq y)$

$\mathbb{P}_2 \cup \text{even}(y)$
Synthesis Problems

Controller Synthesis

Given: process P, specification $\Phi$, and environment E
Find: refinement $P'$ of P so that $P'||E$ satisfies $\Phi$
Solution: $P' =$ winning strategy in game P against E for objective $\Phi$
Synthesis Problems

Controller Synthesis
Given: process P, specification $\Phi$, and environment E
Find: refinement $P'$ of P so that $P'||E$ satisfies $\Phi$
Solution: $P' = \text{winning strategy in game P against E for objective } \Phi$

Co-synthesis
Given:
- two processes $P_1$ and $P_2$
- specifications $\Phi_1$ and $\Phi_2$ for $P_1$ and $P_2$
Find:
refinements $P'_1$ and $P'_2$ of $P_1$ and $P_2$ so that $P'_1||P'_2||S$ satisfies $\Phi_1 - \Phi_2$ for every fair scheduler $S$
Mutual Exclusion

while( true ) {
    flag[1] := true; turn := 2;

    choice
    | while( flag[1] ) nop;
    | while( flag[2] ) nop;
    | while( turn=1 ) nop;
    | while( turn=2 ) nop;
    | while( flag[1] & turn=2 ) nop;
    | while( flag[1] & turn=1 ) nop;
    | while( flag[2] & turn=1 ) nop;
    | while( flag[2] & turn=2 ) nop;
    end choice;

    CS_1 := true; fin_wait;
    CS_1 := false; arbitrary_wait;
}

while( true ) {
    flag[2] := true; turn := 1;

    choice
    | while( flag[1] ) nop;
    | while( flag[2] ) nop;
    | while( turn=1 ) nop;
    | while( turn=2 ) nop;
    | while( flag[1] & turn=2 ) nop;
    | while( flag[1] & turn=1 ) nop;
    | while( flag[2] & turn=1 ) nop;
    | while( flag[2] & turn=2 ) nop;
    end choice;

    CS_2 := true; fin_wait;
    CS_2 := false; arbitrary_wait;
}
Co-synthesis Formulation 1

Do there exist refinements $P'_1$ and $P'_2$ so that $[P'_1 \parallel P'_2 \parallel S] \rightarrow (\Phi_1 - \Phi_2)$ for every fair scheduler $S$?

Solution: game $P_1 \parallel P_2$ against $S$ for objective $\Phi_1 - \Phi_2$

Too weak (solution has $P_1$ and $P_2$ alternate).
Co-synthesis Formulation 2

Do there exist refinements $P'_1$ and $P'_2$ so that both $[P'_1 \parallel P_2 \parallel S] \Rightarrow \Phi_1$ and $[P_1 \parallel P'_2 \parallel S] \Rightarrow \Phi_2$ for every fair scheduler $S$?

Solution: two games $P_1$ against $P_2 \parallel S$ for objective $\Phi_1$, and $P_2$ against $P_1 \parallel S$ for objective $\Phi_2$

Too strong (answer is NO).
Do there exist refinements $P'_1$ and $P'_2$ so that

1. $[P'_1 \parallel P_2 \parallel S] \Rightarrow (\Phi_2 , \Phi_1)$
2. $[P_1 \parallel P'_2 \parallel S] \Rightarrow (\Phi_1 , \Phi_2)$
3. $[P'_1 \parallel P'_2 \parallel S] \Rightarrow (\Phi_1 - \Phi_2)$

for every fair scheduler $S$?
while( true ) {
    flag[1] := true; turn := 2;
    while( flag[2] & turn=1 ) nop;
    CS₁ := true; fin_wait;
    CS₁ := false; arbitrary_wait;
}

while( true ) {
    flag[2] := true; turn := 1;
    while( flag[1] & turn=2 ) nop;
    CS₁ := true; fin_wait;
    CS₁ := false; arbitrary_wait;
}

Solution is exactly Peterson's mutual-exclusion protocol.
Strategic Synthesis Challenge:
Solve co-synthesis for more than two components.
More and Less Solved
Quantitative Aspects in Language Design

Real time
Probabilistic choice

Reliability and robustness
Average resource consumption
Behavioral metrics
Components and Requirements

R₁, R₂, R₃

C₁, C'₁, C₂, C'₂, C''₂
Correctness = Preorder
Fitness = Metric
Example: Incompatible Requirements

\( (R_1, G_1) \quad (R_2, G_2) \quad \square = (G_1 - G_2) \)

\[
\begin{align*}
R_1r_2 & / G_1g_2 \\
r_1R_2 & / g_1G_2 \\
R_1R_2 & / G_1g_2 \\
r_1r_2 & / g_1g_2
\end{align*}
\]
Example: Incompatible Requirements

\[ (R_1, G_1) \quad (R_2, G_2) \quad = (G_1 - G_2) \]
Simulation Preorder
Simulation Distance

![Diagram demonstrating simulation distance with transitions labeled by symbols and probabilities 1/3 and 1/4.](image-url)
Robustness Distance

2/3

1/3

a

b

b

a
Response Times

4,3,2

4
Response Property
Response Monitor

g, t, x

r

S

r, t, x

g

S
Maximal Response

Value of an infinite run is $\liminf$ of $V$. 

\[
\begin{align*}
V &:= 0 \\
V &:= \max(V, C) \\
V &:= \max(V, C) \\
C &:= 0 \\
C &:= C + 1
\end{align*}
\]
Maximal Response Monitor

\[ V := 0 \]

\[ g, t, x \]

\[ r \]

\[ V := \max(V, S) \]

\[ S \]

\[ V := 0 \]

\[ r, x \]

\[ t \]

\[ V := V + 1 \]

\[ g \]

\[ S \]

is final value of \( V \).
Average Response

\[ \text{avg}(V,C,N) = \frac{V_{(N-1)} + C}{N} \]
Average Response Monitor

\[ V := 0 \]
\[ N := 0 \]

\[ V := \text{avg}(V, S, N) \]
\[ N := N + 1 \]
Nested Weighted Automata

Function on weights instead of registers.
Example

Best maximal response time: 2
Worst maximal response time: 3
Emptiness/universality of (max,inc) automata.

Best average response time: 1.5
Worst average response time: 3
Emptiness/universality of (avg,inc) automata.
Expected maximal response time: 2.5
Expectation of (max,inc) automata.

Expected average response time: 2.25
Expectation of (avg,inc) automata.
# Results on (avg,inc) Automata

<table>
<thead>
<tr>
<th></th>
<th>Nondeterministic</th>
<th>Deterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emptiness</td>
<td>([\text{PSPACE,EXPSPACE}])</td>
<td>([\text{PSPACE,EXPSPACE}])</td>
</tr>
<tr>
<td>Universality</td>
<td>undecidable</td>
<td>([\text{PSPACE,EXPSPACE}])</td>
</tr>
<tr>
<td>Expectation</td>
<td>?</td>
<td>(\text{PTIME})</td>
</tr>
</tbody>
</table>
Quantitative Analysis Challenge:

Find the exact complexity of computing average response time over graphs.
Summary

There are plenty of reactive modeling challenges left, not to speak of practical analysis methods and tools.
Some References

1. Simulation distances
   [CONCUR 2010]

2. Nested weighted automata
   [LICS 2015, 2016]

3. Co-synthesis
   [TACAS 2007]

4. Multiple viewpoints
   [EMSOFT 2008]