A Semantic Foundation for Heterogeneous Specification Formalisms

Gerald Luettgen

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia

http://www.icase.edu/~luettgen

joint work with Rance Cleaveland (SUNY at Stony Brook)
Specification Formalisms for Reactive Systems

- Reactive systems:
  - continuously interact with their environment
  - employed in safety-critical / life-critical applications
  - included in automotive, avionics, or embedded systems

- Specification formalisms:
  - concise notations with mathematically exact semantics
  - supported by academic and commercial design tools
  - operational and assertional paradigms
## Operational vs. Assertional Paradigms

<table>
<thead>
<tr>
<th>Operational Paradigm</th>
<th>Assertional Paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>systems</td>
<td>properties</td>
</tr>
<tr>
<td>automata</td>
<td>logic</td>
</tr>
<tr>
<td>equivalences/preorders</td>
<td>satisfaction relation</td>
</tr>
<tr>
<td>concrete designs</td>
<td>abstract requirements</td>
</tr>
<tr>
<td>process algebra</td>
<td>temporal logics</td>
</tr>
</tbody>
</table>

- Often, each paradigm has been investigated in isolation
- Here, study of a heterogeneous specification formalism
Why *Heterogeneous* Specification Formalisms?

- Different formalisms are tuned for different applications
  - mixture of hardware, software, and off-the-shelf components

- Specifications come from different sources
  - different components are often developed by different teams
  - different engineers have different backgrounds

- System requirements should be refinable to system designs
  - gradual transition from abstract requirements to concrete designs
In practice, system architecture is often known *up-front*:

**Example: Protocol Design**

![Diagram of protocol components]

- **Sender**
  - `always`
  - `send? ♦ (put! until gack?)`
  - `send`

- **Medium**
  - `put`
  - `gack`
  - `pack`

- **Receiver**
  - `get`
  - `pack!`
  - `recv!
  - `recv`

---

**assertional specification in linear temporal logic**

**off-the-shelf component**

**operational spec. as automata**
Refinement of Components

- Behavior of *Sender* is concretized by replacing its temporal formula $\Phi$ with an automaton $A$
- Refinement-step is sound since $A$ is a model of $\Phi$
Expressing Fairness Constraints

- Assume, the behavior of *Medium* is specified as follows:

  ![Diagram showing the behavior of Medium]

  \[\text{pack?} \rightarrow \text{drop, gack!} \quad \text{put?} \rightarrow \text{drop, get!}\]

  (*drop* is an internal action signaling loss)

- Suppose, an engineer excludes infinite message droppings:

  \[\text{RealMedium} = \text{Medium} \quad \text{always(put? } \& \text{ eventually(get!))}\]

  fairness constraint

- How to analyze the protocol with *RealMedium*?
Well-known Interrelations

- **Logical characterizations** of behavioral relations:
  \[ P \models Q \quad \text{if and only if} \quad \forall \text{formulas } \Phi. \quad P \models \Phi \implies Q \models \Phi \]

- **Automata-based model checking** for linear-time logics:
  \[ P \models \Phi \quad \text{if and only if} \quad L_B(P) \subseteq L_B(B\Phi) \]

*Formulas as Büchi automata:*

\[ \text{always}(\text{put?} \bigcirc \text{eventually}(\text{get!})) \]
Observations and Background

- Büchi automata may serve as semantic basis for heterogeneous specification formalisms
- Maximal language inclusion is a refinement preorder that is compatible with model checking

**Lesson learned from process algebra:**
maximal language inclusion is not compositional

i.e., it does not support component-based system refinement
Maximal Language Inclusion

\[ P: \quad a \quad a \quad b \quad c \]

\[ Q: \quad a \quad b \quad c \]

But: when (synchronously) composed with

one obtains ...
**Maximal Language Inclusion**

\[ P: \quad a \quad a \quad b \]

\[ Q: \quad a \quad b \]

\[ \text{max. traces} \]

**However:** fully-abstract trace-containment preorders exist, e.g.,

- failure preorder \([Hoare, Brookes/Roscoe]\)
- **must-testing preorder** \([DeNicola/Hennessy]\)
Challenge & New Technical Results

Extend DeNicola/Hennessy’s must-testing preorder $\preceq_{DHmust}$ to a preorder $\preceq_{must}$ on Büchi automata such that

- $\preceq_{must}$ is a conservative extension of $\preceq_{DHmust}$, i.e.,
  \[ P \preceq_{must} Q \iff P \preceq_{DHmust} Q \]

- $\preceq_{must}$ is compatible with linear-time model checking, i.e.,
  \[ P \models \Phi \iff B\Phi \preceq_{must} P \]

- $\preceq_{must}$ is still compositional and fully-abstract [not shown here]

(for classical automata $P$, $Q$ and linear-time formulas $\Phi$)
Executed traces can be finite or infinite

“Real” infinite traces go through “$\tau$- states” infinitely often
Languages and Divergence

Divergence:
“infinite, invisible behavior”

Language:
“observable labels of traces”

<table>
<thead>
<tr>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa, d</td>
<td>∈ ( L_{\text{fin}}(P) )</td>
</tr>
<tr>
<td>d</td>
<td>∈ ( L_{\text{fmax}}(P) )</td>
</tr>
<tr>
<td>ababab..., c</td>
<td>∈ ( L_B(P) )</td>
</tr>
<tr>
<td>c, ca</td>
<td>∈ ( L_{\text{div}}(P) )</td>
</tr>
</tbody>
</table>
Testing Theory for Büchi Automata

- Follow ideas of DeNicola/Hennessy for testing processes
- Tests are specialized Büchi automata with success (😊) states
- Büchi automata should be ordered wrt. their responses to tests
- Define corresponding preorder(s) on Büchi automata
Run Büchi automata $P$ in synchrony with Büchi test $T$, thus, obtaining a trace $tr$.

- $tr$ is a computation, if
  - $tr$ is finite, or
  - $tr$ is infinite & its projection on $P$ is a Büchi trace

- $tr$ is successful, if
  - $tr$ is a finite computation & contains a “☺-state,” or
  - $tr$ is an infinite computation & its projection on $T$ is a Büchi trace
Examples

Automata:

Test:

- Non-computation: \( \langle 1,7 \rangle \xrightarrow{\tau} \langle 5,7 \rangle \xrightarrow{a} \langle 5,8 \rangle \xrightarrow{b} \langle 5,7 \rangle \ldots \)
- Successful computations: \( \langle 1,7 \rangle \xrightarrow{\tau} \langle 5,7 \rangle \xrightarrow{c} \langle 6,9 \rangle \)
  \( \langle 1,7 \rangle \xrightarrow{\tau} \langle 2,7 \rangle \xrightarrow{a} \langle 4,8 \rangle \xrightarrow{b} \langle 2,7 \rangle \ldots \)
- Unsuccessful computation: \( \langle 1,7 \rangle \xrightarrow{\tau} \langle 2,7 \rangle \xrightarrow{b} \langle 2,7 \rangle \ldots \)
May and Must Preorders

- Interpretation of Success:
  - $P \text{ may } T$ means some computation is successful (failure is tolerated)
  - $P \text{ must } T$ means all computations are successful (failure is catastrophic)

- May-testing: $P \mathcal{Q}_{\text{may}} Q$ if $\forall T. P \text{ may } T \Rightarrow Q \text{ may } T$

- Must-testing: $P \mathcal{Q}_{\text{must}} Q$ if $\forall T. P \text{ must } T \Rightarrow Q \text{ must } T$

Research questions:

- What are alternative characterizations?
- What is the relationship to traditional testing?
Alternative Characterizations

**Theorem.** ($P$ and $Q$ are Büchi automata)

- $P \Delta_{\text{may}} Q \iff L_{\text{fin}}(P) \subseteq L_{\text{fin}}(Q)$ and $L_B(P) \subseteq L_B(Q)$

- $P \Delta_{\text{must}} Q \iff \forall w \not\in L_{\text{div}}(P)$.
  
  (a) $w \not\in L_{\text{div}}(Q)$
  
  (b) $w$ finite $\Rightarrow$

  $$\forall Q'. Q \xrightarrow{w} Q' \Rightarrow \exists P'. P \xrightarrow{w} P' \land I(P') \subseteq I(Q')$$

  (c) $w \in L_B(Q) \Rightarrow w \in L_B(P)$

*initial action set inclusion*
Conservative Extensions

- DeNicola/Hennessy’s testing theory (\(\mathcal{D}_{DHmust}, \mathcal{D}_{DHmay}\)) has been developed for labeled transition systems (LTS)

- LTS = Büchi automata where every state is a “\(\square\) - state”

- For LTSs \(P\) and \(Q\):
  \[ P \mathcal{D}_{must} Q \iff P \mathcal{D}_{DHmust} Q \]

- For non-divergent & image-finite LTSs \(P\) and \(Q\):
  \[ P \mathcal{D}_{may} Q \iff P \mathcal{D}_{DHmay} Q \]
Relationship to Linear-time Logic

**Goal:**
"
Φ ⊩ must \( P \) ⇔ \( P |\models \Phi \)
"

**Approach:**

- Introduce concept of pure nondeterminism
- Show for Büchi automata \( N \) and \( P \) (for \( N \) purely nondeterministic)

\[
N \models_{\text{must}} P \iff L(P) \subseteq L(N)
\]

- Construct for any formula \( \Phi \) in linear-time logic a purely nondeterministic Büchi automaton \( N_\Phi \) such that

\[
L(P) \subseteq L(N_\Phi) \iff P |\models \Phi
\]
Purely Nondeterministic Büchi Automata

**Observe:**
- \( \mathcal{L} \) satisfies \( a \ until \ b \) but not \( D \ \Box_{\text{must}} \ b \)
- Choices may not be made by tests but by properties!
**Theorem.** Let $N$ and $P$ such that $N$ is purely nondeterministic. Then $N \models_{\text{must}} P$ if and only if

1. $L_{\text{div}}(P) \subseteq L_{\text{div}}(N)$
2. $L_{\text{fin}}(P) - L_{\text{div}}(N) \subseteq L_{\text{fin}}(N)$
3. $L_{\text{fmax}}(P) - L_{\text{div}}(N) \subseteq L_{\text{fmax}}(N)$
4. $L_B(P) - L_{\text{div}}(N) \subseteq L_B(N)$

→ Büchi must-testing of purely nondeterministic Büchi automata reduces to language inclusion
Linear-time Logics and
Purely Nondeterministic Büchi Automata

Fact. [cf. Vardi/Wolper] For every formula $\Phi$ in linear-time logic, a purely nondeterministic Büchi automaton $N_\Phi$ can be constructed such that for all $P$:

$$P \models \Phi \iff \text{Equations (1)-(4) hold for } P \text{ and } N_\Phi$$

Example. (divergence states represent tautologies)

a until b:
Synthesis and Conclusions

Corollary. \[ P \models \Phi \iff N_\Phi \bigcirc_{\text{must}} P \]

- A semantic theory uniting
  - refinement based on must-testing for Büchi automata, &
  - (compositional) model checking of linear-time logics (LTL)

In essence:

- Must-testing + nondeterminism = model checking for LTL

Alternative approach: [not investigated here]

- Must-testing = model checking for \( \mathcal{G} \text{LTL} + \text{“} \bigcirc \text{I”-atoms} \)
Future Work

- Developing languages for heterogeneous specifications
  - combining process algebras and linear-time logics
- Investigating their algebraic properties
  - congruence properties, fully-abstractness, and axiomatizations
- Deriving decision procedures for Büchi testing preorders
  - implementation of the testing preorders in a verification tool
- Conducting case studies
  - future-generation flight guidance systems
  - communication protocols for data buses used in avionics systems
Related Work

- Modular model checking [Kupferman/Vardi]
- Partial Model Checking [Andersen/Winskel, Larsen]
- Fair testing [Natarajan/Cleaveland]
- Büchi testing [Kumar/Cleaveland/Smolka]
- Implicit Specifications [Larsen]
Thanks!