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

Operational Paradigm	***Assertional Paradigm***
systems automata equivalences/preorders	properties logic satisfaction relation
concrete designs	abstract requirements
process algebra	temporal logics

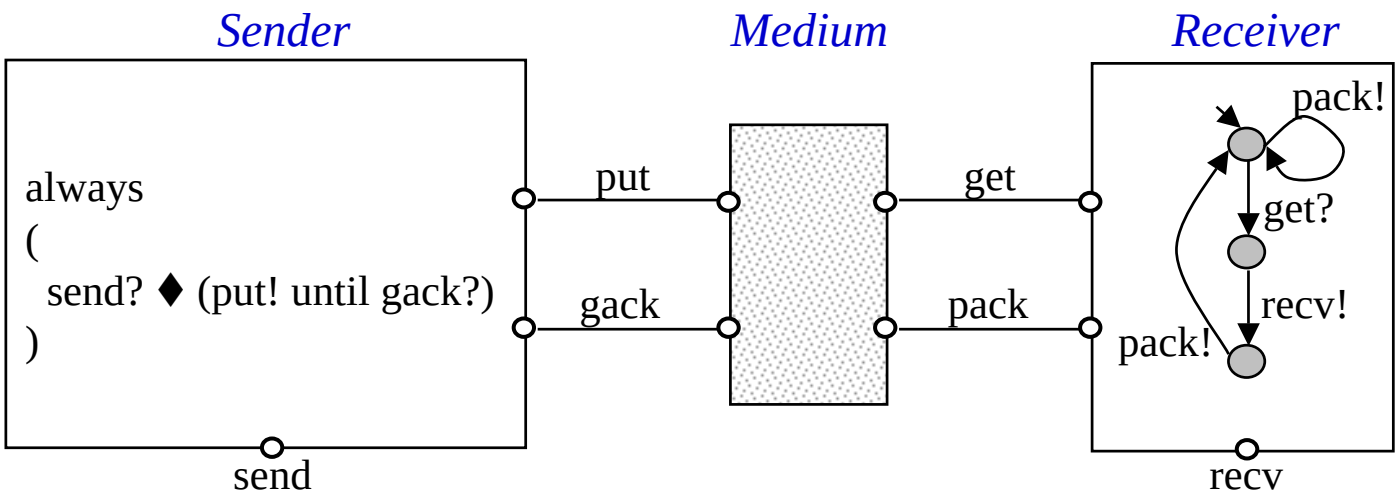
- Often, each paradigm has been investigated in isolation
- Here, study of a heterogeneous specification formalism

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

Example: Protocol Design

In practice, system architecture is often known *up-front*:



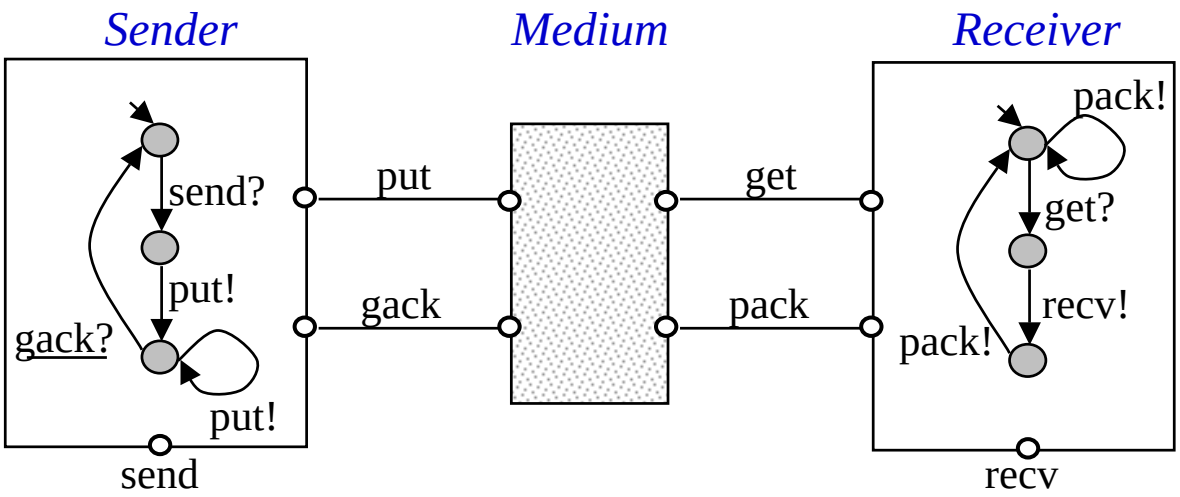
assertional specification in linear temporal logic

off-the-shelf component

operational spec. as automata

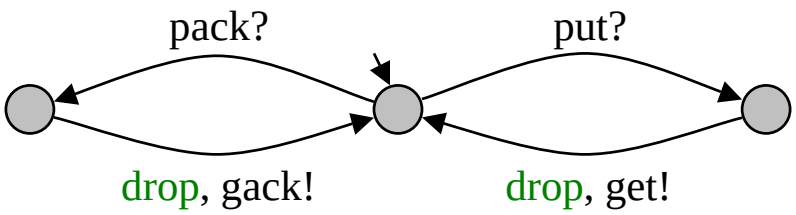
Refinement of Components

- Behavior of *Sender* is concretized by replacing its temporal formula Φ with an automaton A
- Refinement-step is sound since A *is a model of* Φ



Expressing Fairness Constraints

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



(drop is an internal action signaling loss)

- Suppose, an engineer excludes infinite message droppings:

$$\mathit{RealMedium} = \mathit{Medium} \ \underline{\mathit{always}(\text{put?} \blacklozenge \mathit{eventually}(\text{get!}))}$$

fairness constraint

- How to analyze the protocol with *RealMedium*?

Well-known Interrelations

- Logical characterizations of behavioral relations:

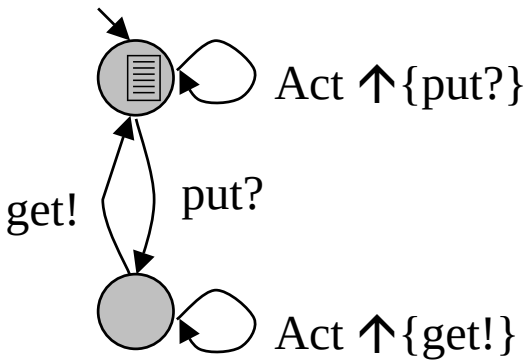
$$P \mathrel{\wr\!\wr} Q \quad \text{if and only if} \quad \forall \text{formulas } \Phi.\ P \models \Phi \ \Rightarrow\ Q \models \Phi$$

- 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:

always(put? $\mathrel{\underline{\wr\!\wr}}$ *eventually*(get!))



Observations and Background

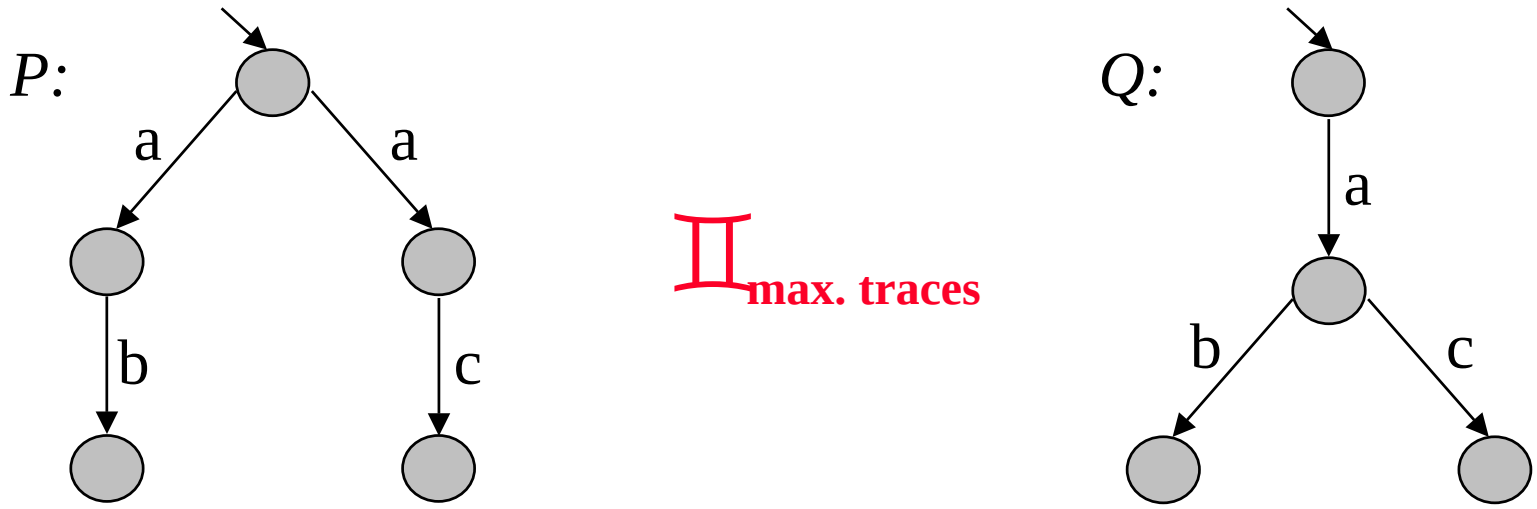
- Büchi automata may serve as semantic basis for heterogeneous specification formalisms
- Maximal language inclusion is a refinement preorder that is compatible with model checking

***Lesson learned from process algebra*:**

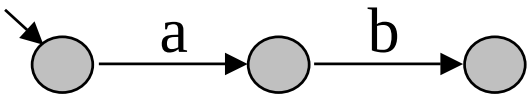
maximal language inclusion is <u>not compositional</u>

i.e., it does <u>not</u> support component-based system refinement

Maximal Language Inclusion

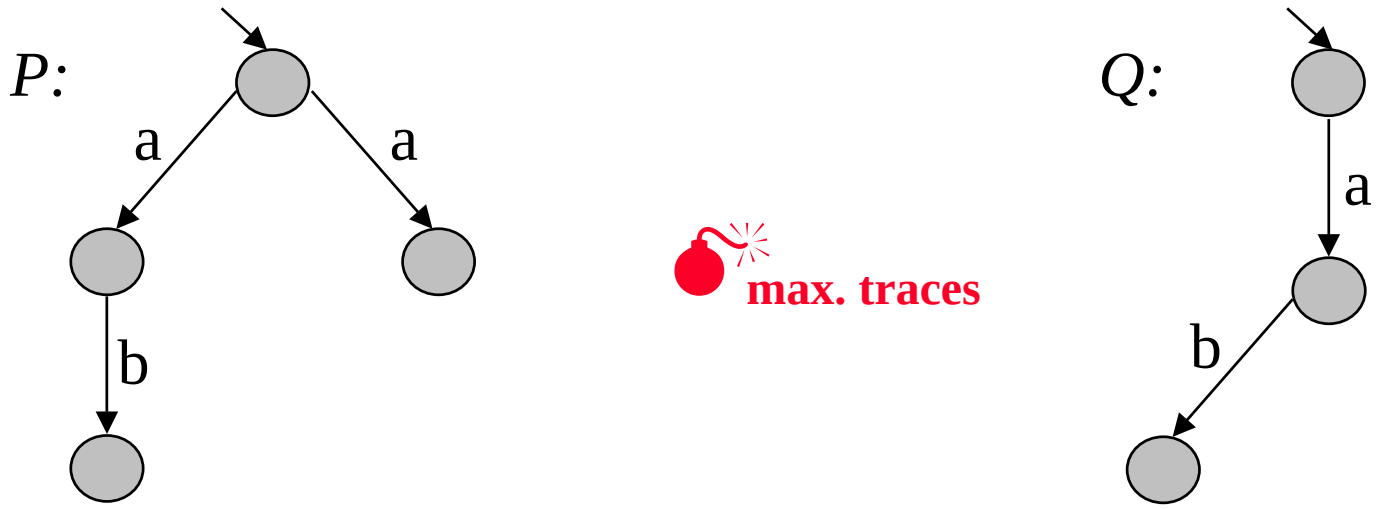


But: when (synchronously) composed with



one obtains ...

Maximal Language Inclusion



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 $\sqsubseteq_{\mathrm{DHmust}}$ to a preorder $\sqsubseteq_{\mathrm{must}}$ on Büchi automata such that

- $\sqsubseteq_{\mathrm{must}}$ is a conservative extension of $\sqsubseteq_{\mathrm{DHmust}}$, i.e.,

$$P \sqsubseteq_{\mathrm{must}} Q \quad \Leftrightarrow \quad P \sqsubseteq_{\mathrm{DHmust}} Q$$

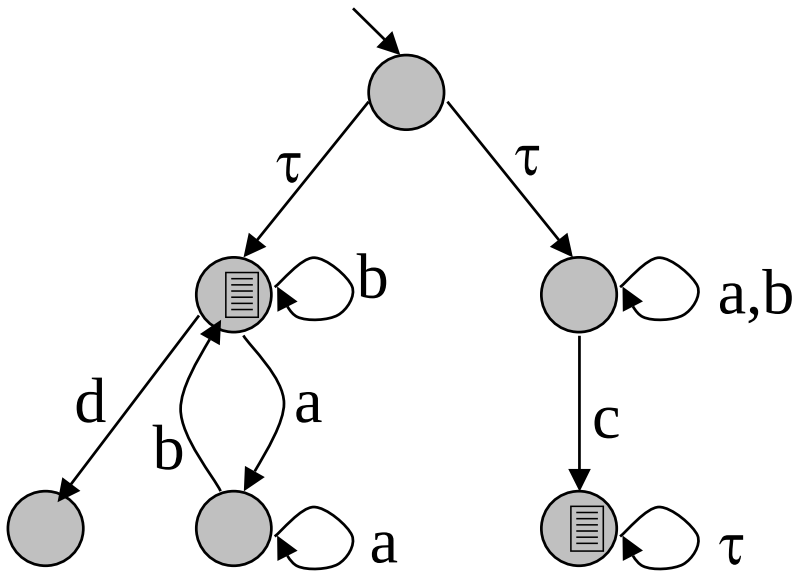
- $\sqsubseteq_{\mathrm{must}}$ is compatible with linear-time model checking, i.e.,

$$P \models \Phi \quad \Leftrightarrow \quad \mathrm{B}_{\Phi} \sqsubseteq_{\mathrm{must}} P$$

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

(for *classical* automata P, Q and linear-time formulas Φ)

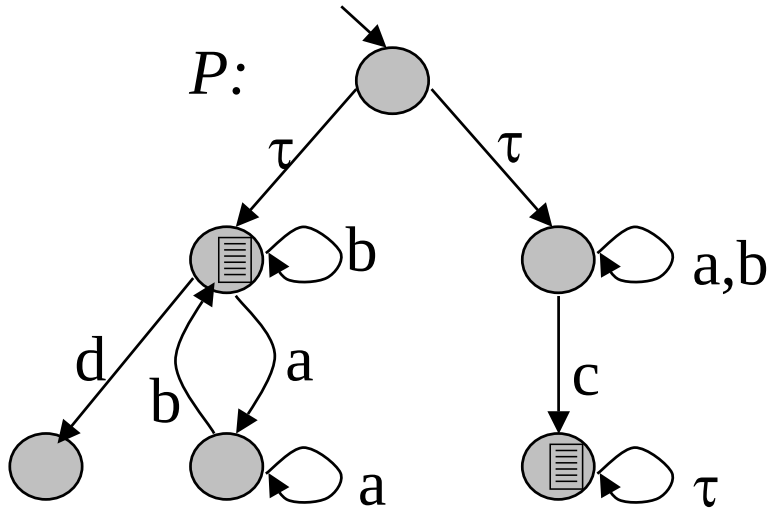
Introduction to B•chi Automata



(τ denotes a distinguished, internal action)

- Executed traces can be finite or infinite
- "Real" infinite traces go through "- states" infinitely often

Languages and Divergence



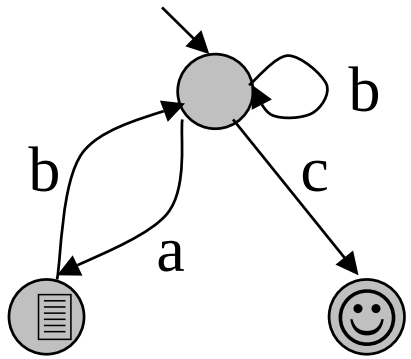
Divergence:
"infinite, invisible behavior"

Language:
"observable labels of traces"

aa, d	$\in L_{fin}(P)$	(finite language)
d	$\in L_{fmax}(P)$	(finite maximal language)
ababab..., c	$\in L_{B}(P)$	(Büchi language)
c, ca	$\in L_{div}(P)$	(divergent language)

Testing Theory for B•chi Automata

- Follow ideas of *DeNicola/Hennessy* for testing processes
- Tests are specialized Büchi automata with success (☺) states



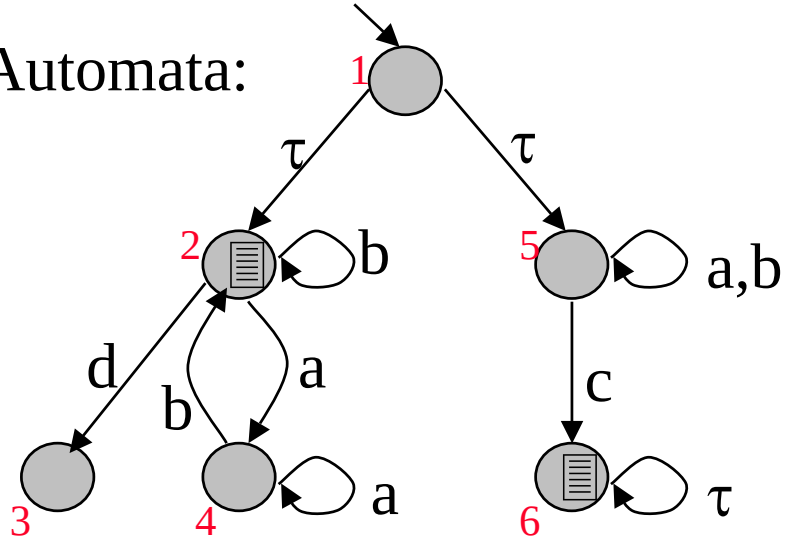
- Büchi automata should be ordered wrt. their responses to tests
- Define corresponding preorder(s) on Büchi automata

Computation and Success

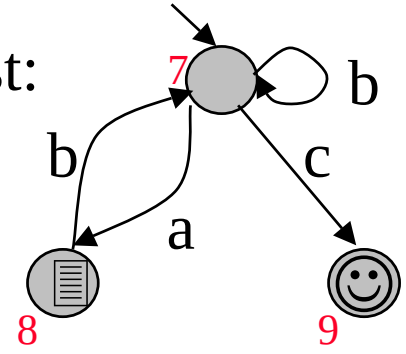
- Run Büchi automata *P* in synchrony with Büchi test *T*, thus, obtaining a trace *tr*
- *tr* is a computation, if
  - *tr* is finite, or
  - *tr* is infinite & its projection on *P* is a Büchi trace
- *tr* is successful, if
  - *tr* is a finite computation & contains a "☺-state," or
  - *tr* is an 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* may *T*** means <u>some</u> computation is successful (failure is tolerated)
  - ***P* must *T*** means <u>all</u> computations are successful (failure is catastrophic)
- May-testing: $P \sqsubseteq_{\text{may}} Q$ if $\forall T.\ P \text{ may } T \Rightarrow Q \text{ may } T$
- Must-testing: $P \sqsubseteq_{\text{must}} Q$ 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)

- $P \, \delta\!\!\!\wr_{may} \, Q$ iff $L_{fin}(P) \subseteq L_{fin}(Q)$ and $\underline{L_B(P) \subseteq L_B(Q)}$

- $P \, \delta\!\!\!\wr_{must} \, Q$ iff $\forall w \notin L_{div}(P)$.

  (a) $w \notin L_{div}(Q)$

  (b) w finite $\Rightarrow$

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

  (c) $\underline{w \in L_B(Q) \;\Rightarrow\; w \in L_B(P)}$

initial action set inclusion

Conservative Extensions

- DeNicola/Hennessy's testing theory ($\sqsubseteq_{DHmay}$, $\sqsubseteq_{DHmust}$) has been developed for labeled transition systems (LTS)
- LTS = Büchi automata where every state is a "$\Box$- state"
- For LTSs P and Q:

$$P \sqsubseteq_{must} Q \quad \text{iff} \quad P \sqsubseteq_{DHmust} Q$$

- For non-divergent & image-finite LTSs P and Q:

$$P \sqsubseteq_{may} Q \quad \text{iff} \quad P \sqsubseteq_{DHmay} Q$$

Relationship to Linear-time Logic

***Goal*:** $$"\Phi \ \mathfrak{S}_{\text{must}} \ P \quad \Leftrightarrow \quad P \models \Phi"$$

***Approach*:**

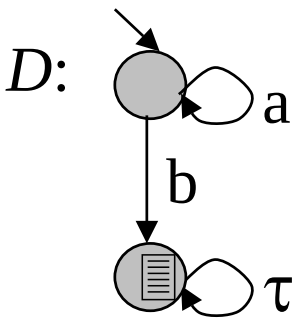
- Introduce concept of pure nondeterminism
- Show for Büchi automata N and P (for N purely nondeterministic)

$$N \ \mathfrak{S}_{\text{must}} \ P \quad \Leftrightarrow \quad \mathrm{L}(P) \subseteq \mathrm{L}(N)$$

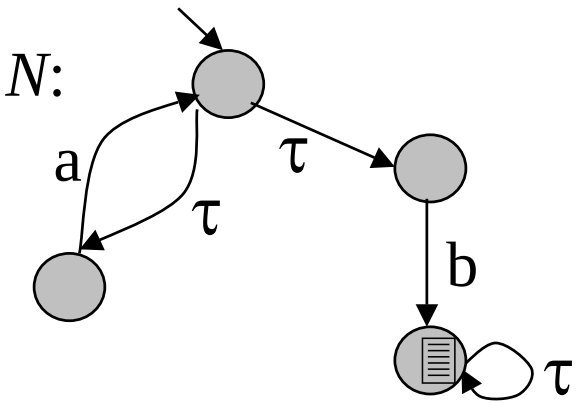
- Construct for any formula Φ in linear-time logic a purely nondeterministic Büchi automaton N_Φ such that

$$\mathrm{L}(P) \subseteq \mathrm{L}(N_\Phi) \quad \Leftrightarrow \quad P \models \Phi$$

Purely Nondeterministic Büchi Automata



a *until* b



deterministic | temporal property | purely nondeterministic

Observe:

- $\xrightarrow{b}$ satisfies a *until* b but not $D \, \Omega_{must} \xrightarrow{b}$
- Choices may not be made by tests but by properties!

Büchi Must-Testing and Purely Nondeterministic Büchi Automata

Theorem. Let N and P such that N is purely nondeterministic.

Then $N \; \delta_{\text{must}} \; P$ if and only if

$$(1) \quad L_{div}(P) \subseteq L_{div}(N)$$

$$(2) \quad L_{fin}(P) - L_{div}(N) \subseteq L_{fin}(N)$$

$$(3) \quad L_{fmax}(P) - L_{div}(N) \subseteq L_{fmax}(N)$$

$$(4) \quad L_{B}(P) - L_{div}(N) \subseteq L_{B}(N)$$

➡ Büchi must-testing of purely nondeterministic Büchi automata reduces to language inclusion

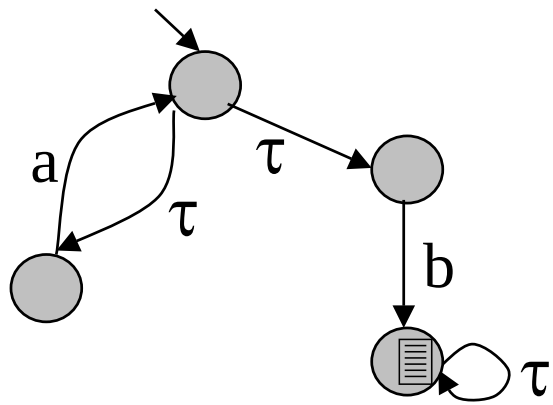
Linear-time Logics and Purely Nondeterministic Büchi Automata

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

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

Example. (divergence states represent tautologies)

a *until* b:



Synthesis and Conclusions

Corollary. $P \models \Phi \quad \Leftrightarrow \quad N_\Phi \sqsubseteq_{\text{must}} P$

- 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 $\mathcal{P}$LTL + "Π I"-atoms $\mathcal{N}$

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*!**