A Foundation for Runtime Monitoring

The Secret Life of Monitors

Adrian Francalanza · March 20, 2018

More than one way to skin a cat

How do you ensure that your program is correct?

More than one way to skin a cat

How do you ensure that your program is correct?

Verification Techniques

- Testing
- Model Checking
- Type Checking
- Theorem Proving
- ...

What if...

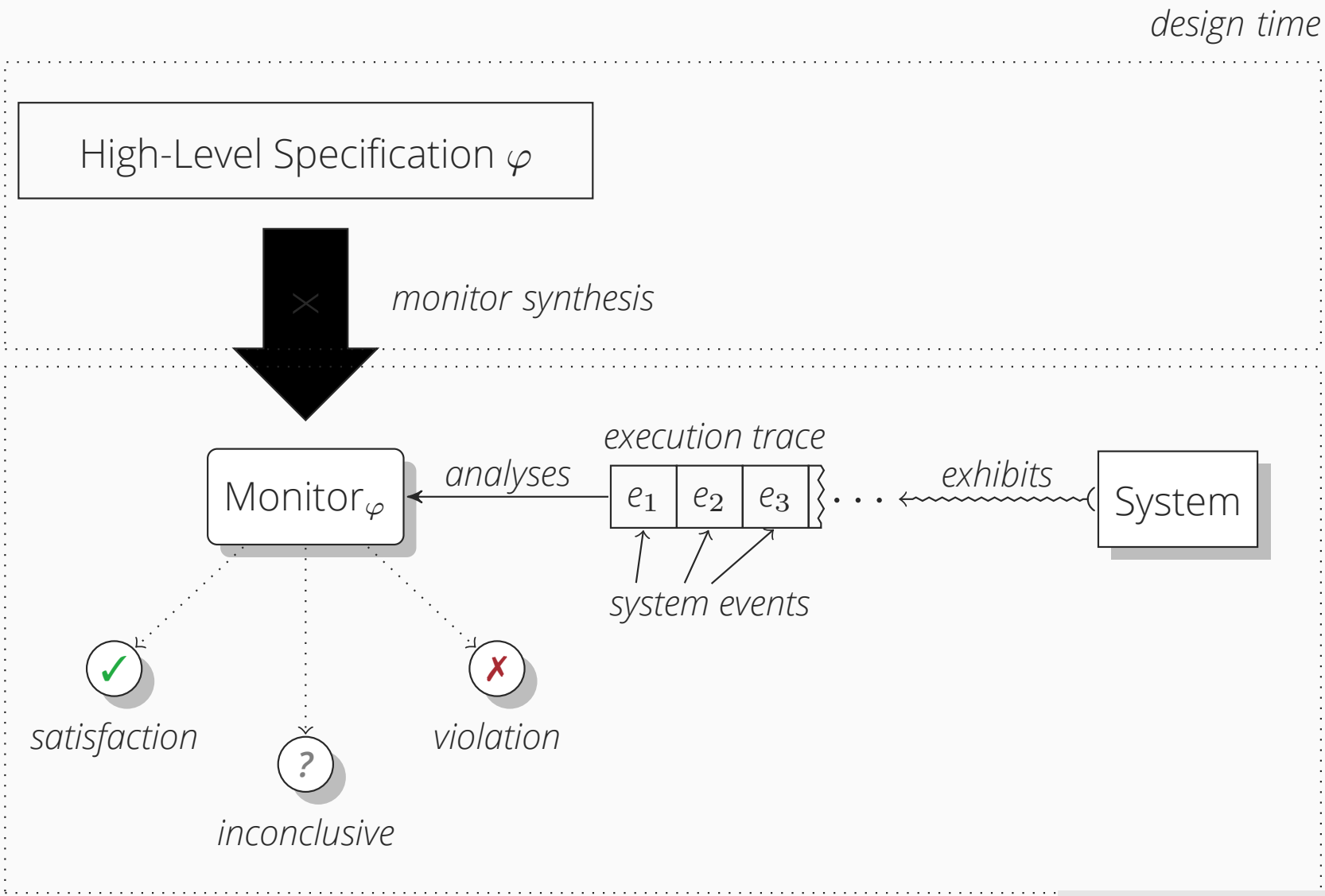
...Building a model is not worthwhile?

- Building a model is very **expensive** and/or **complex**.
- The model may need to **change often**.

...Building a model is not possible?

- The system should be viewed as a **black box**.
- Parts of the system are third-party or **closed source**.
- Parts of the system are **dynamically** loaded at runtime.

The General Approach



Why Runtime Monitoring

- a code-structuring technique (design pattern).
- an incremental/compositional method of development.

Why Runtime Monitoring

- a code-structuring technique (design pattern).
- an incremental/compositional method of development.
- a lightweight verification method.

Why Runtime Monitoring

- a code-structuring technique (design pattern).
- an incremental/compositional method of development.
- a lightweight verification method.
- a post-deployment verification method.

State of Affairs

Why Runtime Monitoring

- a code-structuring technique (design pattern).
- an incremental/compositional method of development.
- a lightweight verification method.
- a post-deployment verification method.

Where is Monitoring used in Practice?

How: In conjunction with other verification techniques.

Where: Autonomous Vehicles, Financial Transactions, Online Video Games, Machine Learning *etc.*

Main Concerns in Runtime Verification (RV)

1. Efficiency
2. Expressivity
3. Correctness

State of Affairs

Main Concerns in Runtime Verification (RV)

1. Efficiency
2. **Expressivity**
3. Correctness

Main Concerns in Runtime Verification (RV)

1. Efficiency
2. Expressivity
3. **Correctness**

State of Affairs

Main Concerns in Runtime Verification (RV)

1. Efficiency
2. Expressivity
3. Correctness

The Aims of Our Study

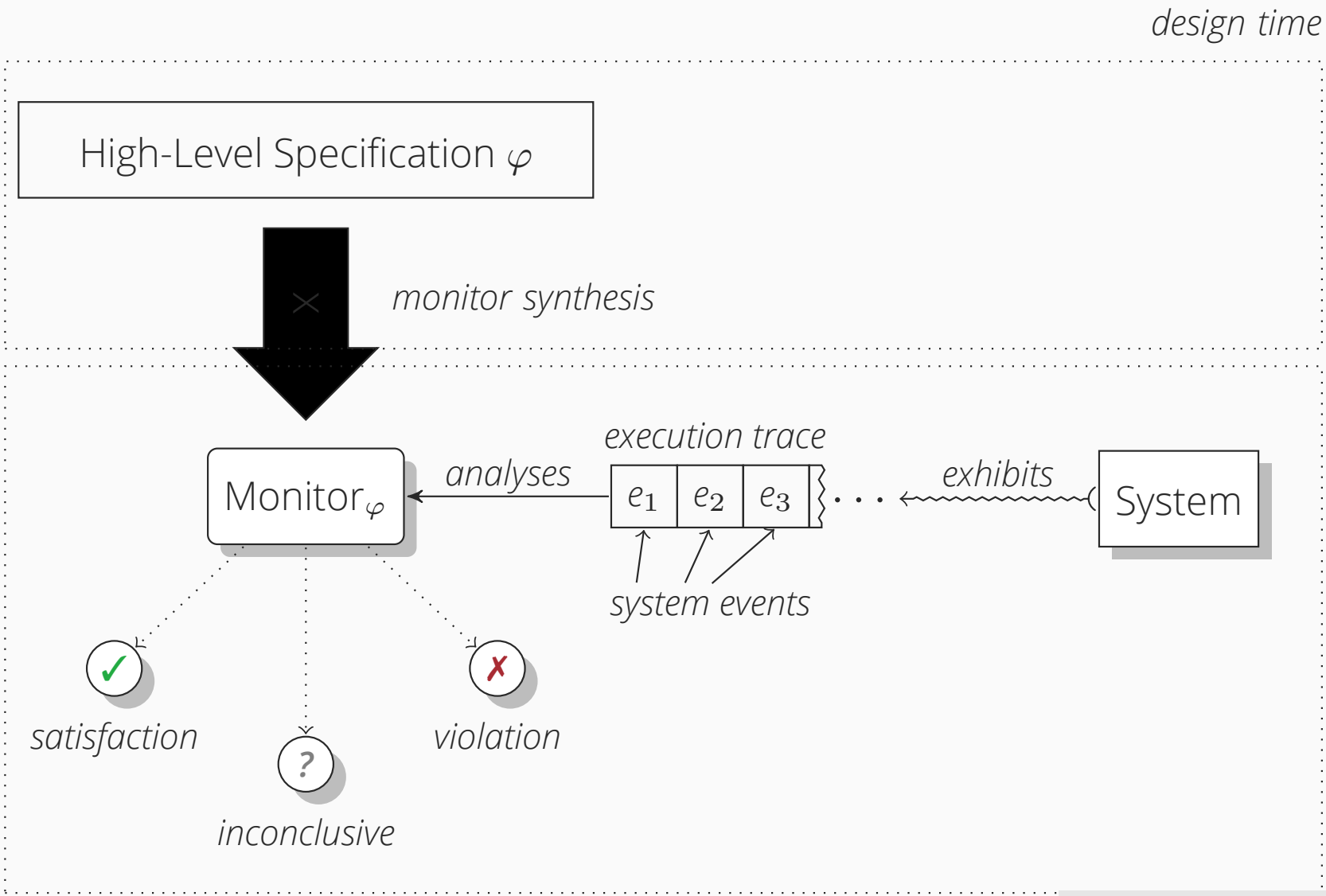
- Understand the **expressivity limits** of RV.
- Identify **correctness criteria** and design **methods** to **ensure correctness**.

Outline

1. Preliminaries
2. Monitors and Instrumentation
3. Monitorability
4. To Monitorabilty...and Beyond!

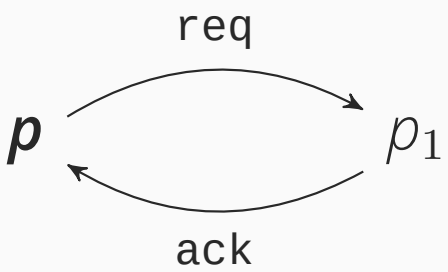
Preliminaries

The General Approach

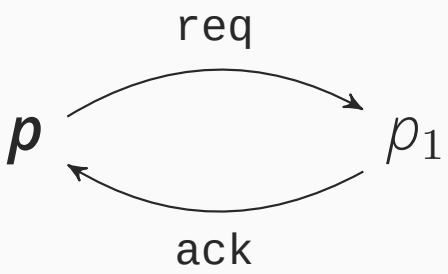


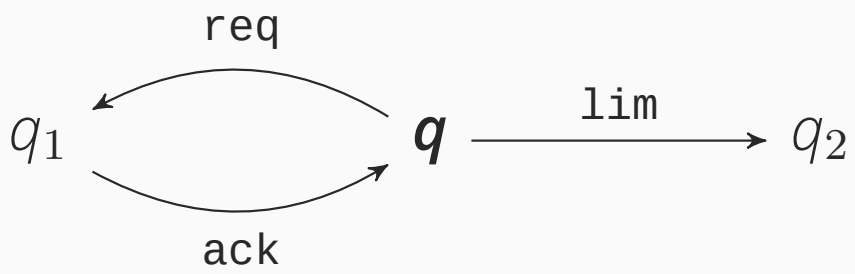
The Model

The Model

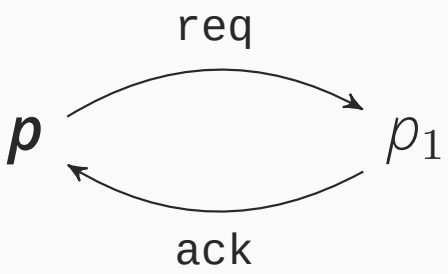


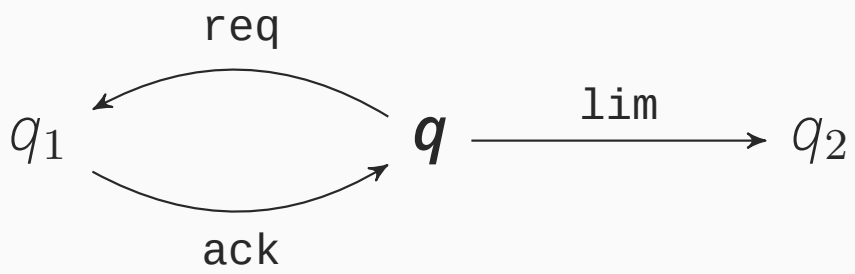
The Model

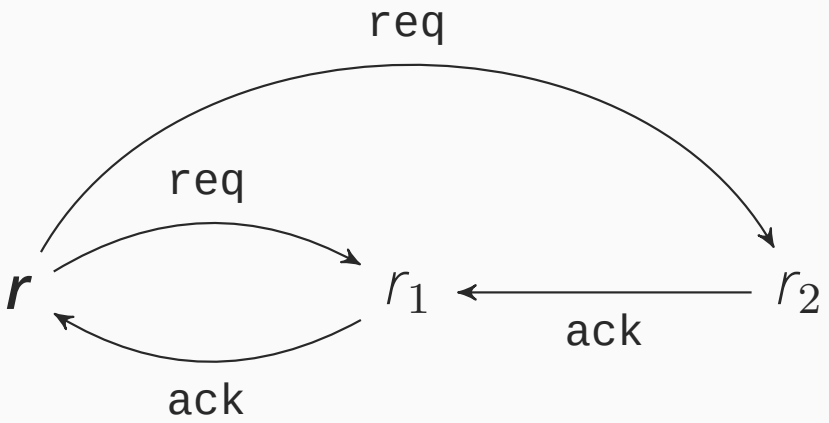




The Model







The Model

Definition (Labelled Transition Systems (LTS))

An LTS is a triple

$$\langle \text{Sys}, (\text{Act} \cup \{\tau\}), \longrightarrow \rangle$$

consisting of:

The Model

Definition (Labelled Transition Systems (LTS))

An LTS is a triple

$$\langle \text{Sys}, \ (\text{Act} \cup \{\tau\}), \ \longrightarrow \rangle$$

consisting of:

- a set of **system states** $p, q, r \in \text{Sys}$,

The Model

Definition (Labelled Transition Systems (LTS))

An LTS is a triple

$$\langle \text{Sys}, \ (\text{Act} \cup \{\tau\}), \ \longrightarrow \rangle$$

consisting of:

- a set of **system states** $p, q, r \in \text{Sys}$,
- a set of **visible actions** $\alpha \in \text{Act}$ and a distinguished **silent action** τ, where $\tau \notin \text{Act}$ and μ ranges over $(\text{Act} \cup \{\tau\})$,

The Model

Definition (Labelled Transition Systems (LTS))

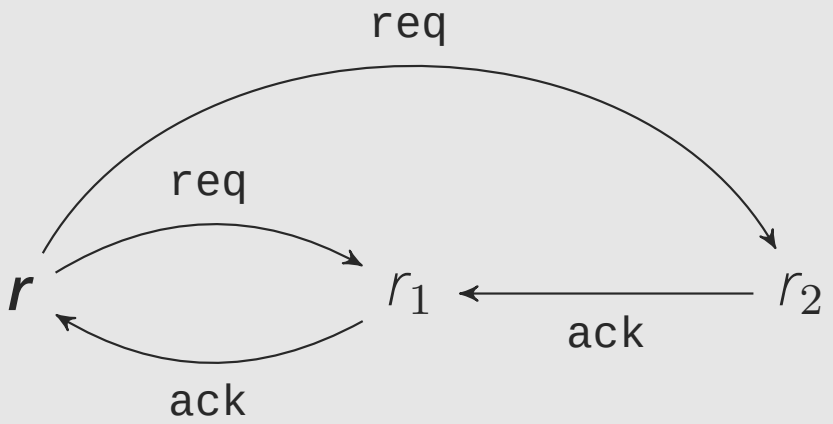
An LTS is a triple

$$\langle \text{Sys}, \ (\text{Act} \cup \{\tau\}), \ \longrightarrow \rangle$$

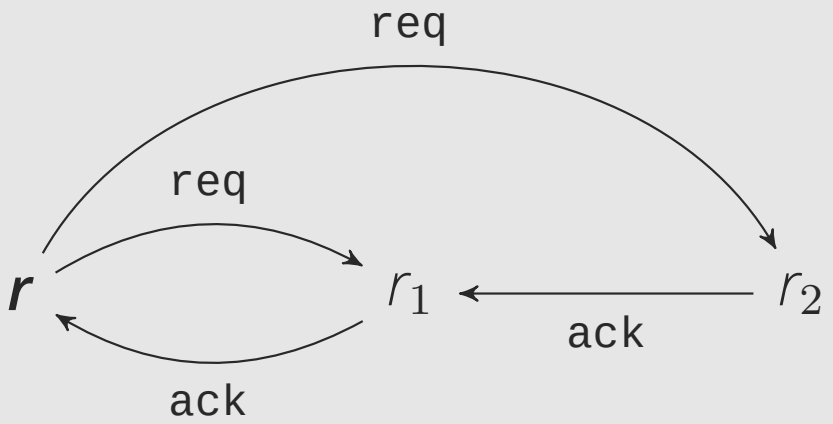
consisting of:

- a set of **system states** $p, q, r \in \text{Sys}$,
- a set of **visible actions** $\alpha \in \text{Act}$ and a distinguished **silent action** τ, where $\tau \notin \text{Act}$ and μ ranges over $(\text{Act} \cup \{\tau\})$,
- a ternary **transition relation** between states labelled by actions; we write $p \xrightarrow{\alpha} q$ in lieu of $\langle p, \alpha, q \rangle \in \longrightarrow$.

Systems as LTS



Systems as LTS



$$\left\langle \begin{array}{l} \{r, r_1, r_2\},\ \{\text{req}, \text{ack}, \tau\} \\ \quad \{r \xrightarrow{\text{req}} r_1,\ r \xrightarrow{\text{req}} r_2,\ r_1 \xrightarrow{\text{ack}} r,\ r_2 \xrightarrow{\text{ack}} r_1\} \end{array} \right\rangle$$

Quiz

Automata and Labelled Transition Systems

What is the difference between an LTS and an Automaton?

Automata and Labelled Transition Systems

What is the difference between an LTS and an Automaton?

- An automaton must have a **finite** number of states.
- An automaton must have a **finite** number of labels.

Automata and Labelled Transition Systems

What is the difference between an LTS and an Automaton?

- An automaton must have a **finite** number of states.
- An automaton must have a **finite** number of labels.
- An LTS does **not** have final states (in our case — there are some LTSs variants that have terminal configurations).
- We typically assume rooted LTS so both have a unique start state.

Automata and Labelled Transition Systems

What is the difference between an LTS and an Automaton?

- An automaton must have a **finite** number of states.
- An automaton must have a **finite** number of labels.
- An LTS does **not** have final states (in our case — there are some LTSs variants that have terminal configurations).
- We typically assume rooted LTS so both have a unique start state.
- LTSs have a more explicit **delineation** between **internal** and **external** actions (and transitions).
- Automata usually need to specify **explicitly** that their transitions are **non-deterministic**.

The Logic

The Logic

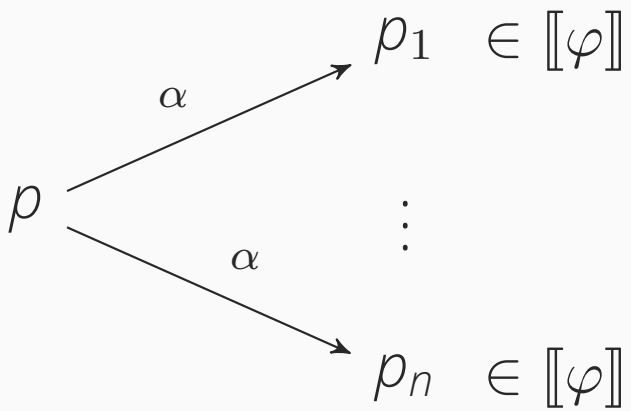
Definition (Logic Syntax)

$$
\begin{array}{rcll}
\varphi, \phi \in \mathit{HML} ::= & & \mathbf{ff} & \text{(falsehood)} \\
& | & \mathbf{tt} & \text{(truth)} \\
& | & \varphi \wedge \phi & \text{(conjunction)} \\
& | & \varphi \vee \phi & \text{(disjunction)} \\
& | & [\alpha]\varphi & \text{(necessity)} \\
& | & \langle\alpha\rangle\varphi & \text{(possibility)}
\end{array}
$$

The Logic

Definition (Logic Syntax)

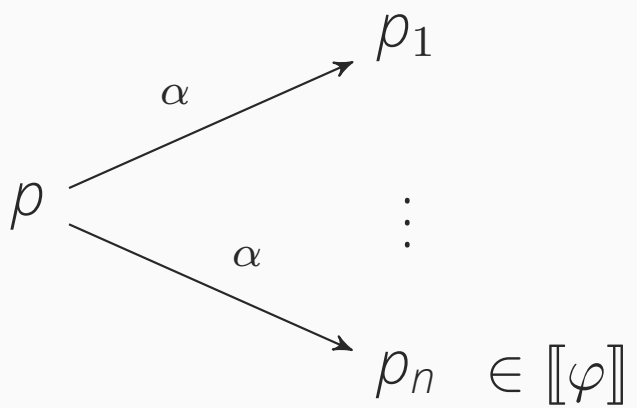
$$
\begin{array}{rcll}
\varphi, \phi \in \mathit{HML} ::= & \mathbf{ff} & & \text{(falsehood)} \\
\mid & \mathbf{tt} & & \text{(truth)} \\
\mid & \varphi \wedge \phi & & \text{(conjunction)} \\
\mid & \varphi \vee \phi & & \text{(disjunction)} \\
\mid & [\alpha]\varphi & & \text{(necessity)} \\
\mid & \langle\alpha\rangle\varphi & & \text{(possibility)}
\end{array}
$$



The Logic

Definition (Logic Syntax)

$$
\begin{array}{rcll}
\varphi, \phi \in \mathit{HML} ::= & \mathbf{ff} & & \text{(falsehood)} \\
\mid & \mathbf{tt} & & \text{(truth)} \\
\mid & \varphi \wedge \phi & & \text{(conjunction)} \\
\mid & \varphi \vee \phi & & \text{(disjunction)} \\
\mid & [\alpha]\varphi & & \text{(necessity)} \\
\mid & \langle\alpha\rangle\varphi & & \text{(possibility)}
\end{array}
$$



Definition (Logic Semantics)

$$
\begin{aligned}
[\![\mathbf{ff}]\!] &\stackrel{\text{def}}{=} \emptyset \\
[\![\mathbf{tt}]\!] &\stackrel{\text{def}}{=} \text{Sys} \\
[\![\varphi \wedge \phi]\!] &\stackrel{\text{def}}{=} [\![\varphi]\!] \cap [\![\phi]\!] \\
[\![\varphi \vee \phi]\!] &\stackrel{\text{def}}{=} [\![\varphi]\!] \cup [\![\phi]\!] \\
[\![[\alpha]\varphi]\!] &\stackrel{\text{def}}{=} \left\{ p \mid \forall p'.\ p \xrightarrow{\alpha} p' \text{ implies } p' \in [\![\varphi]\!] \right\} \\
[\![\langle\alpha\rangle\varphi]\!] &\stackrel{\text{def}}{=} \left\{ p \mid \exists p'.\ p \xrightarrow{\alpha} p' \text{ and } p' \in [\![\varphi]\!] \right\}
\end{aligned}
$$

Quiz

What does this Formula Mean?

$$\langle \alpha \rangle \mathbf{tt}$$

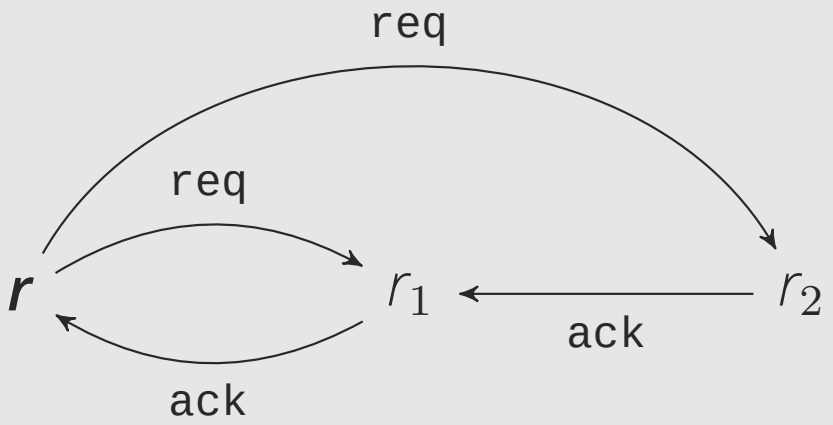
Definition (Logic Semantics)

$$
\begin{aligned}
[\![\mathbf{ff}]\!] &\stackrel{\text{def}}{=} \emptyset \\
[\![\mathbf{tt}]\!] &\stackrel{\text{def}}{=} \text{Sys} \\
[\![\varphi \wedge \phi]\!] &\stackrel{\text{def}}{=} [\![\varphi]\!] \cap [\![\phi]\!] \\
[\![\varphi \vee \phi]\!] &\stackrel{\text{def}}{=} [\![\varphi]\!] \cup [\![\phi]\!] \\
[\![[\alpha]\varphi]\!] &\stackrel{\text{def}}{=} \left\{ p \mid \forall p' .\ p \xrightarrow{\alpha} p' \text{ implies } p' \in [\![\varphi]\!] \right\} \\
[\![\langle \alpha \rangle \varphi]\!] &\stackrel{\text{def}}{=} \left\{ p \mid \exists p' .\ p \xrightarrow{\alpha} p' \text{ and } p' \in [\![\varphi]\!] \right\}
\end{aligned}
$$

What does this Formula Mean?

$$\langle \alpha \rangle \mathbf{tt}$$

Core Formulas



$r \in [\![\langle \text{req} \rangle \mathbf{tt}]\!]$ $\quad$ $r_1 \notin [\![\langle \text{req} \rangle \mathbf{tt}]\!]$ $\quad$ $r_2 \notin [\![\langle \text{req} \rangle \mathbf{tt}]\!]$

What does this Formula Mean?

$$[\beta]\langle\alpha\rangle\mathbf{tt}$$

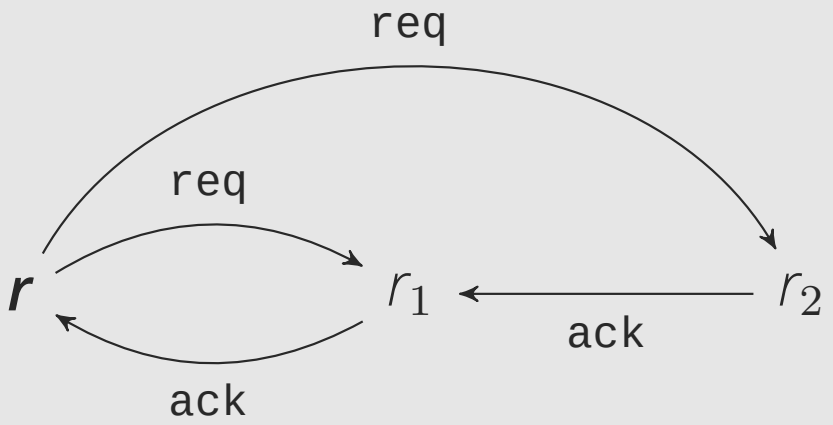
Definition (Logic Semantics)

$$
\begin{aligned}
[\![\mathbf{ff}]\!] &\stackrel{\text{def}}{=} \emptyset \\
[\![\mathbf{tt}]\!] &\stackrel{\text{def}}{=} \text{Sys} \\
[\![\varphi \wedge \phi]\!] &\stackrel{\text{def}}{=} [\![\varphi]\!] \cap [\![\phi]\!] \\
[\![\varphi \vee \phi]\!] &\stackrel{\text{def}}{=} [\![\varphi]\!] \cup [\![\phi]\!] \\
[\![[\alpha]\varphi]\!] &\stackrel{\text{def}}{=} \left\{ p \mid \forall p' .\ p \xrightarrow{\alpha} p' \text{ implies } p' \in [\![\varphi]\!] \right\} \\
[\![\langle\alpha\rangle\varphi]\!] &\stackrel{\text{def}}{=} \left\{ p \mid \exists p' .\ p \xrightarrow{\alpha} p' \text{ and } p' \in [\![\varphi]\!] \right\}
\end{aligned}
$$

What does this Formula Mean?

$$[\beta]\langle\alpha\rangle\mathbf{tt}$$

More on Core Formulas

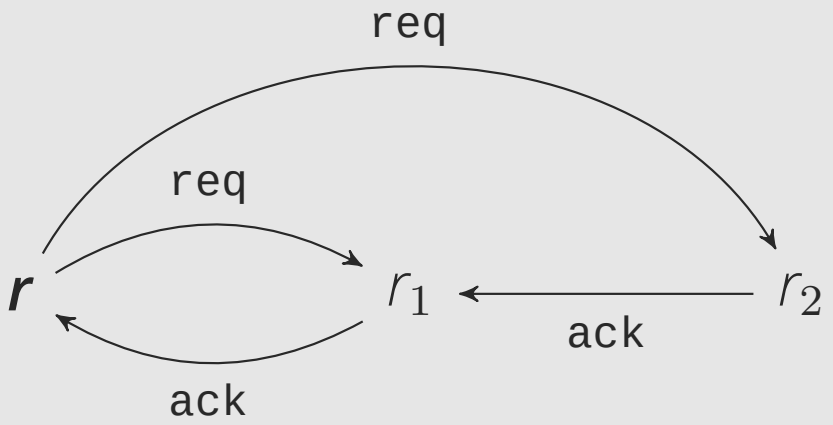


$r \in [\![[\text{req}]\langle\text{ack}\rangle\mathbf{tt}]\!]$

What does this Formula Mean?

$$[\beta]\langle\alpha\rangle\mathbf{tt}$$

More on Core Formulas

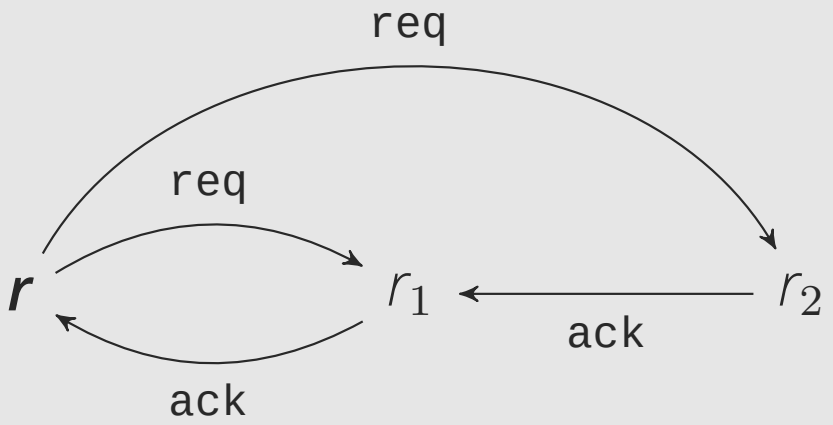


$r \in [\![[\text{req}]\langle\text{ack}\rangle\mathbf{tt}]\!]$ r_1 ?? $[\![[\text{req}]\langle\text{ack}\rangle\mathbf{tt}]\!]$ r_2 ?? $[\![[\text{req}]\langle\text{ack}\rangle\mathbf{tt}]\!]$

What does this Formula Mean?

$$[\beta]\langle\alpha\rangle\mathbf{tt}$$

More on Core Formulas



$r \in [\![[\text{req}]\langle\text{ack}\rangle\mathbf{tt}]\!]$ $r_1 \in [\![[\text{req}]\langle\text{ack}\rangle\mathbf{tt}]\!]$ $r_2 \in [\![[\text{req}]\langle\text{ack}\rangle\mathbf{tt}]\!]$

What does this Formula Mean?

$$[\alpha]\mathbf{ff}$$

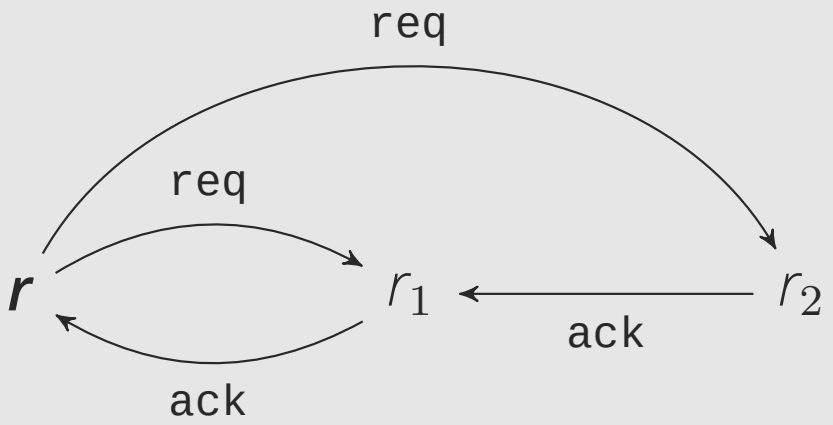
Definition (Logic Semantics)

$$
\begin{aligned}
[\![\mathbf{ff}]\!] &\stackrel{\text{def}}{=} \emptyset \\
[\![\mathbf{tt}]\!] &\stackrel{\text{def}}{=} \text{Sys} \\
[\![\varphi \wedge \phi]\!] &\stackrel{\text{def}}{=} [\![\varphi]\!] \cap [\![\phi]\!] \\
[\![\varphi \vee \phi]\!] &\stackrel{\text{def}}{=} [\![\varphi]\!] \cup [\![\phi]\!] \\
[\![[\alpha]\varphi]\!] &\stackrel{\text{def}}{=} \left\{ p \mid \forall p' .\ p \xrightarrow{\alpha} p' \text{ implies } p' \in [\![\varphi]\!] \right\} \\
[\![\langle\alpha\rangle\varphi]\!] &\stackrel{\text{def}}{=} \left\{ p \mid \exists p' .\ p \xrightarrow{\alpha} p' \text{ and } p' \in [\![\varphi]\!] \right\}
\end{aligned}
$$

What does this Formula Mean?

$$[\alpha]\mathbf{ff}$$

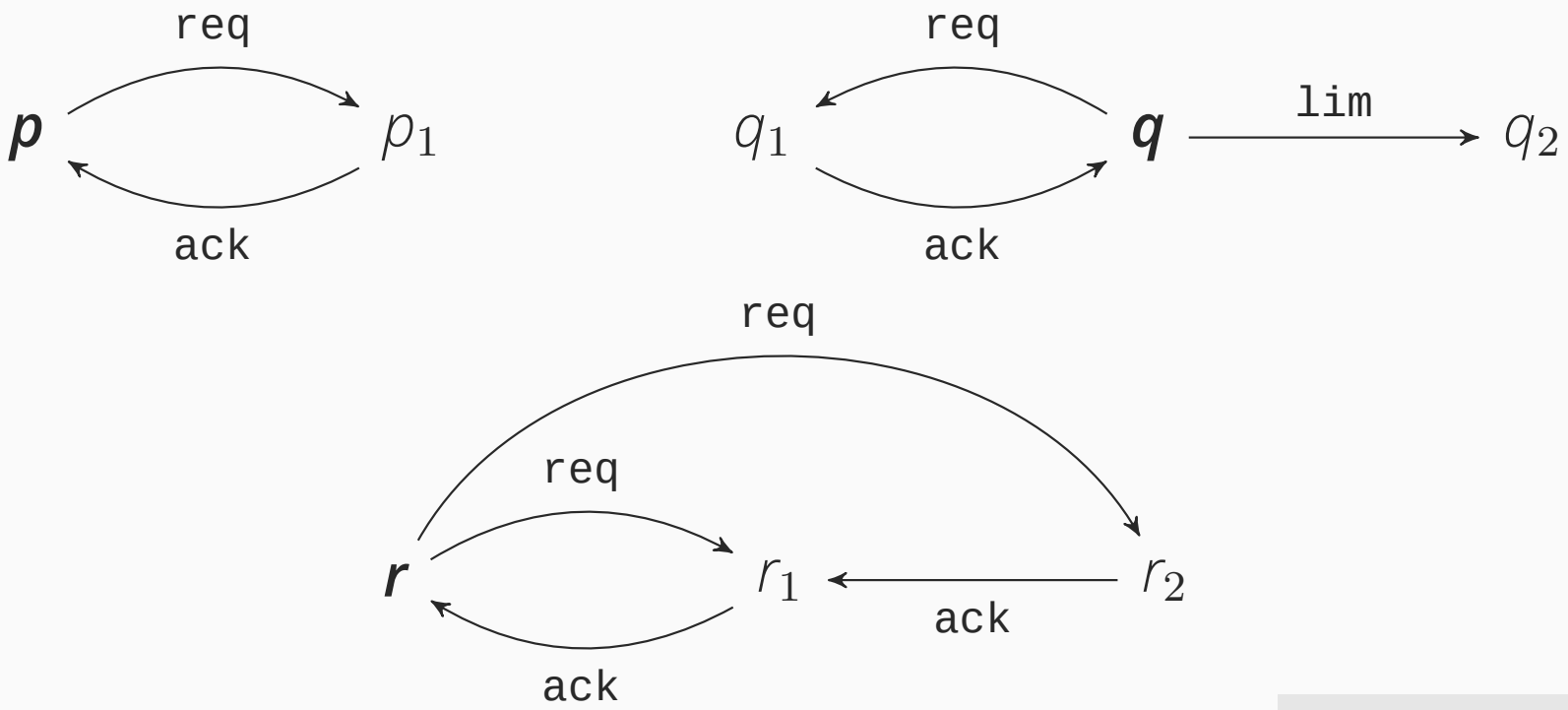
Core Formulas



$r \notin [\![[\text{req}]\mathbf{ff}]\!]$ $\quad$ $r_1 \in [\![[\text{req}]\mathbf{ff}]\!]$ $\quad$ $r_2 \in [\![[\text{req}]\mathbf{ff}]\!]$

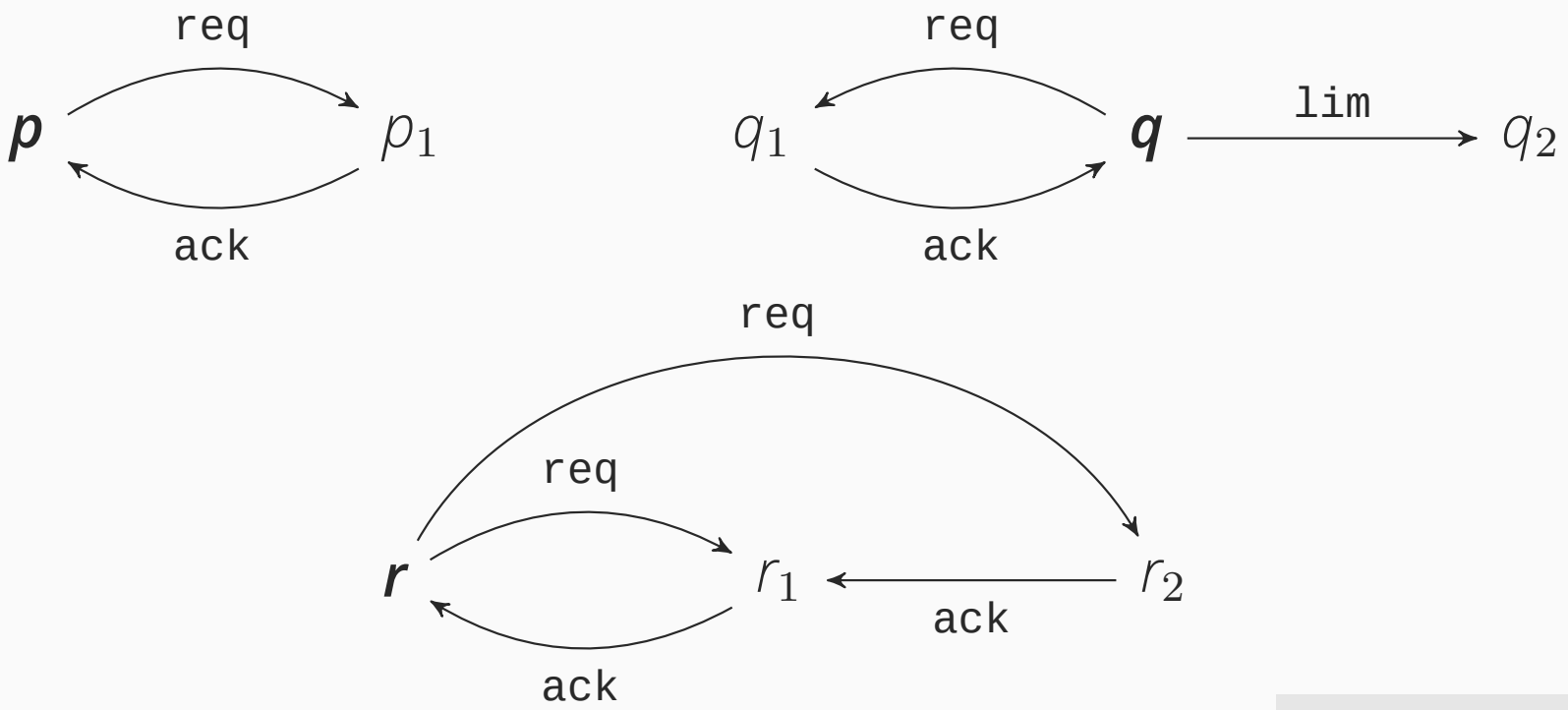
System Properties

$$\langle \texttt{req} \rangle \langle \texttt{ack} \rangle \langle \texttt{req} \rangle \mathbf{tt}$$



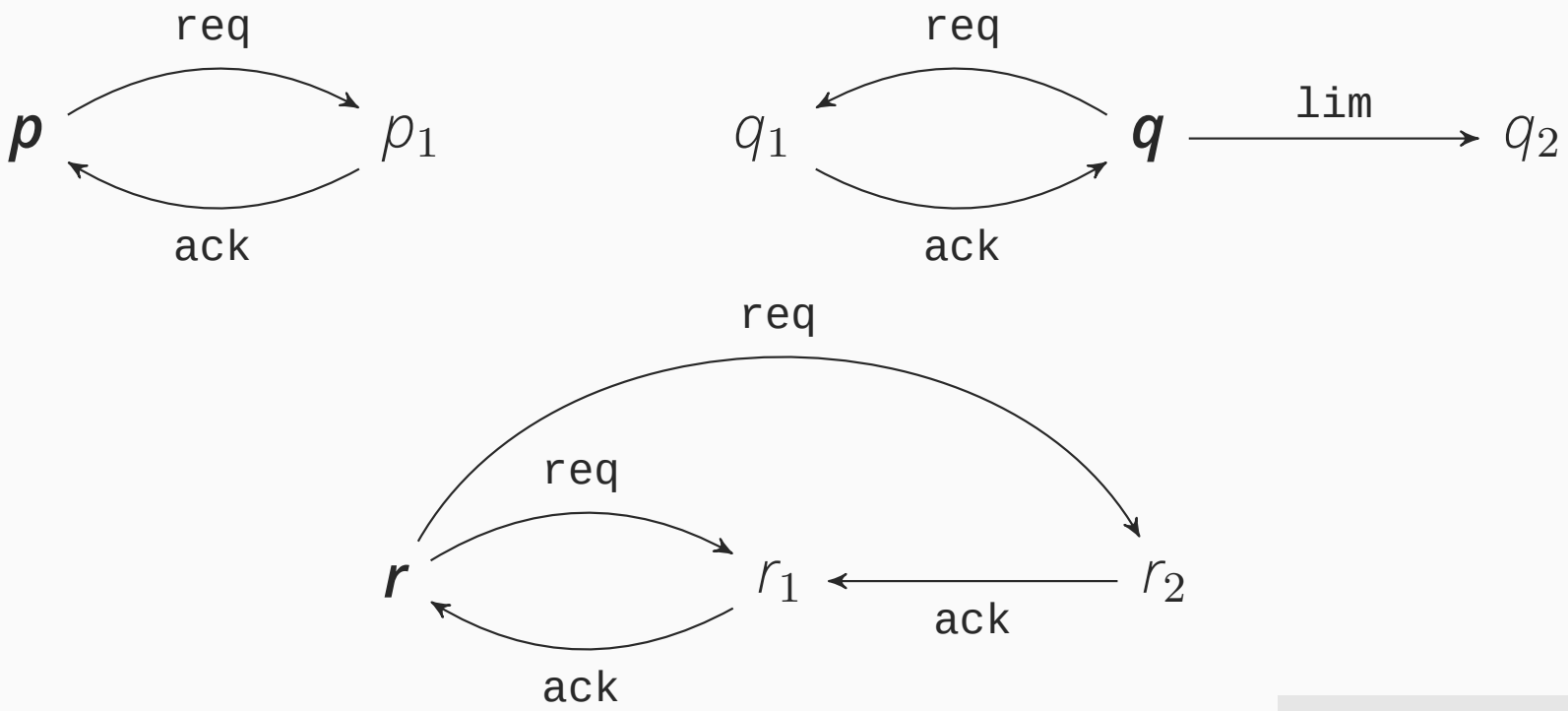
System Properties

$$p, q, r \in [\![\langle \texttt{req} \rangle \langle \texttt{ack} \rangle \langle \texttt{req} \rangle \textbf{tt}]\!]$$



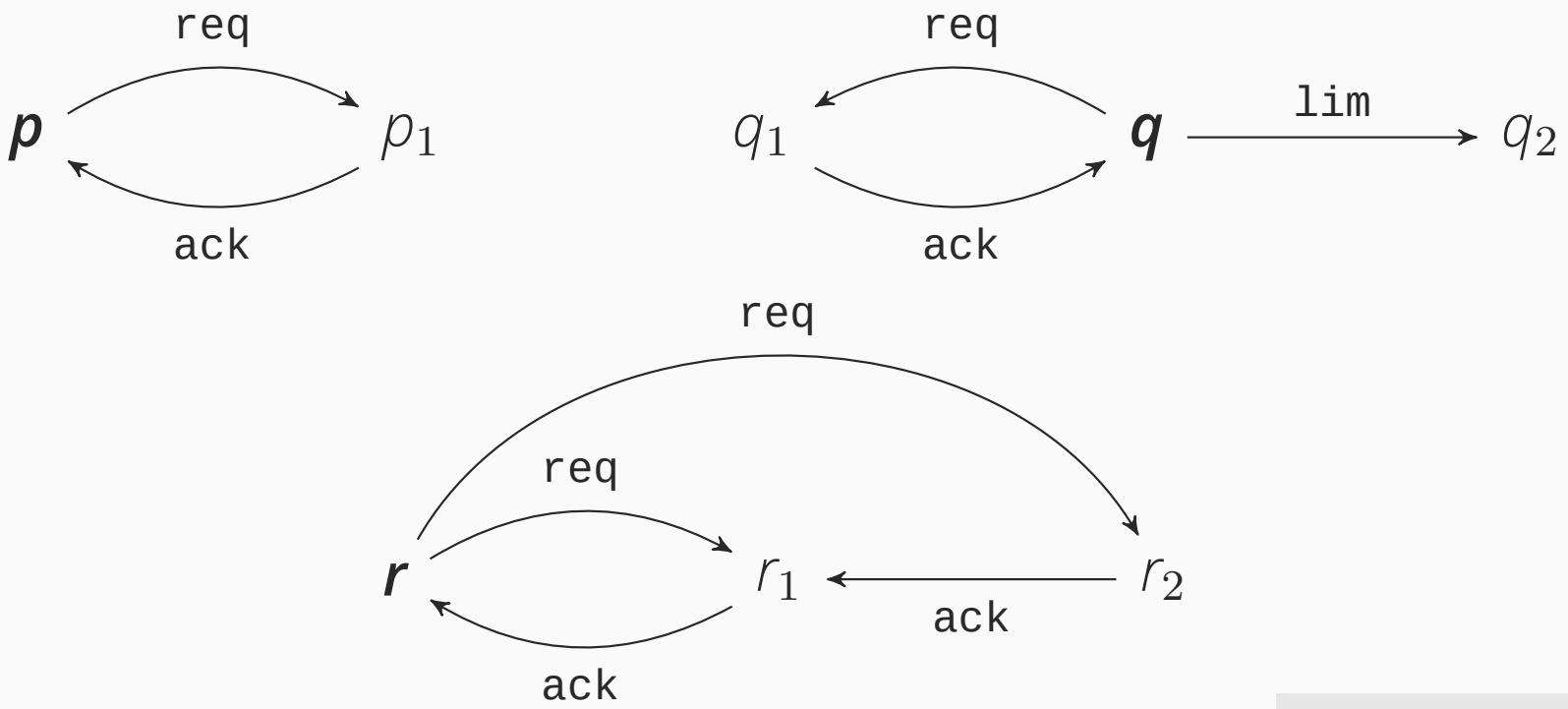
System Properties

$$[\texttt{req}][\texttt{ack}]\langle\texttt{req}\rangle\mathbf{tt}$$



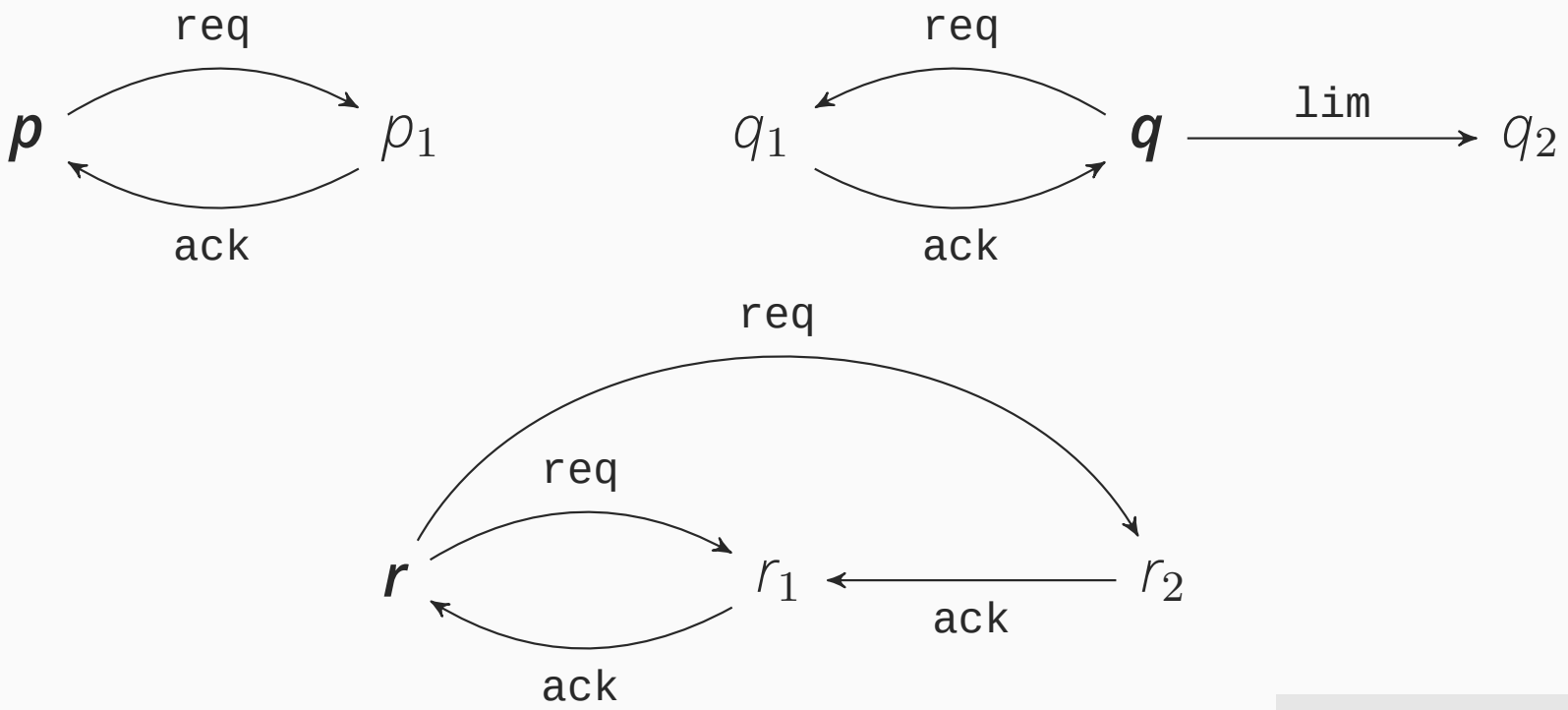
System Properties

$$p, q \in [\![\, [\texttt{req}][\texttt{ack}]\langle\texttt{req}\rangle \mathbf{tt} \,]\!]$$



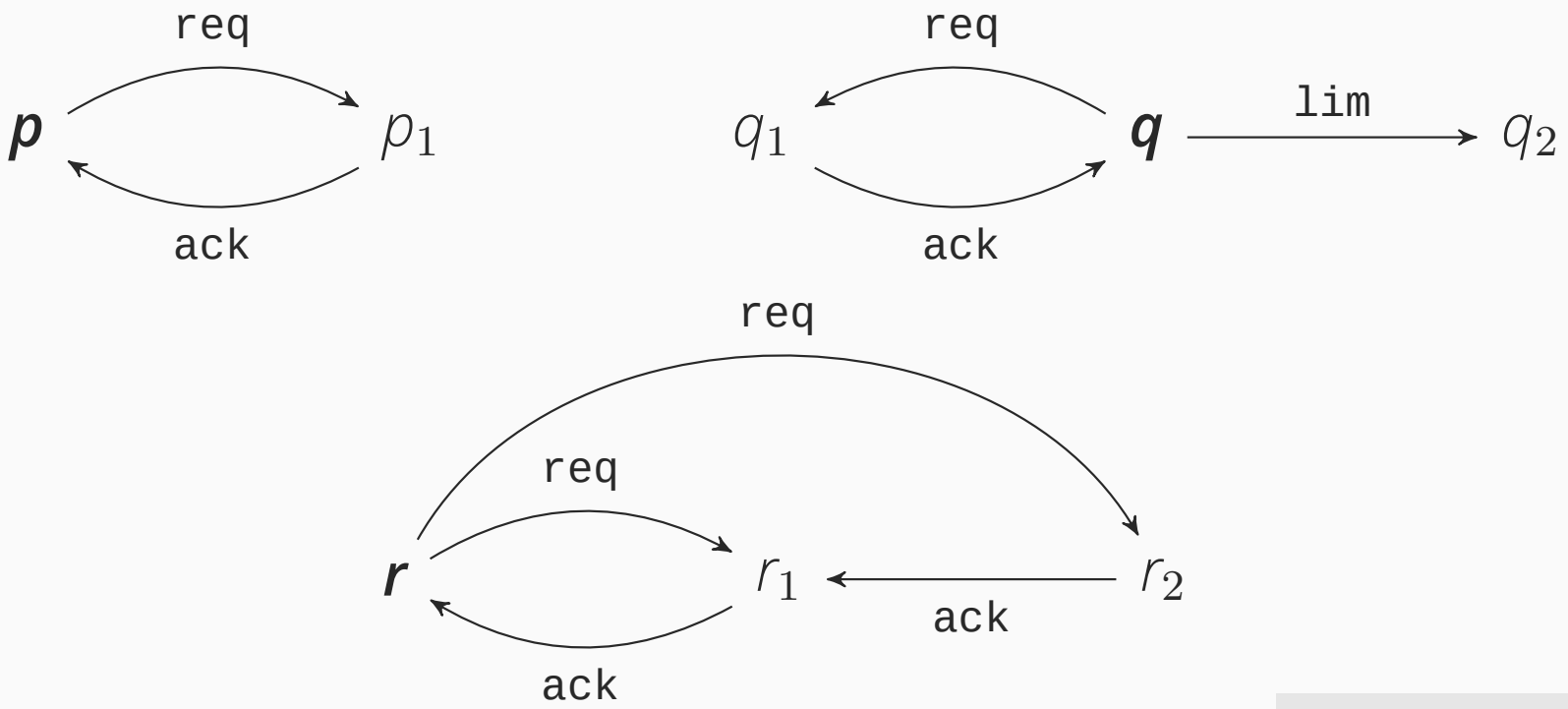
System Properties

$$[\texttt{req}][\texttt{ack}]\langle\texttt{req}\rangle\mathbf{tt} \wedge \langle\texttt{lim}\rangle\mathbf{tt}$$



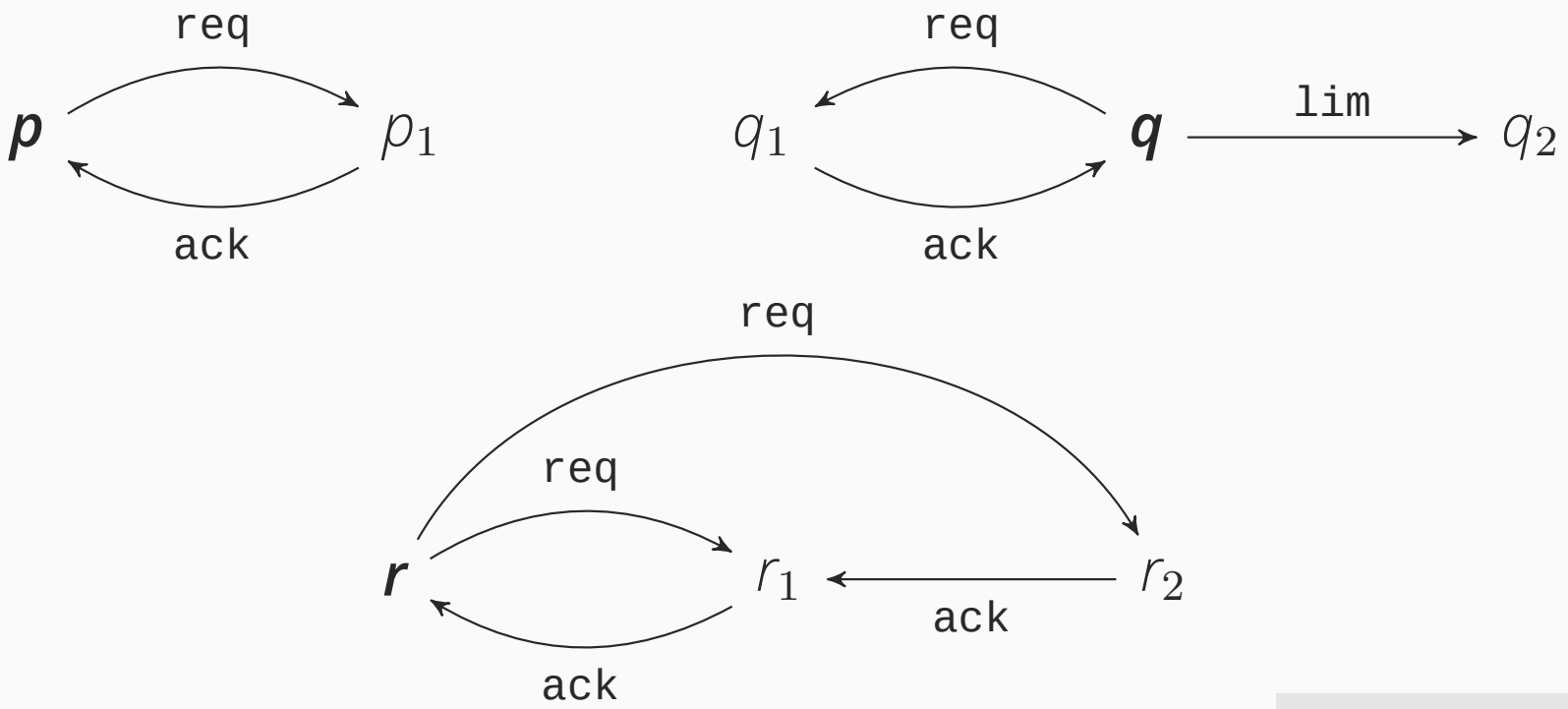
System Properties

$$q \in [\![\, [\texttt{req}][\texttt{ack}]\langle \texttt{req}\rangle \mathbf{tt} \wedge \langle \texttt{lim}\rangle \mathbf{tt} \,]\!]$$



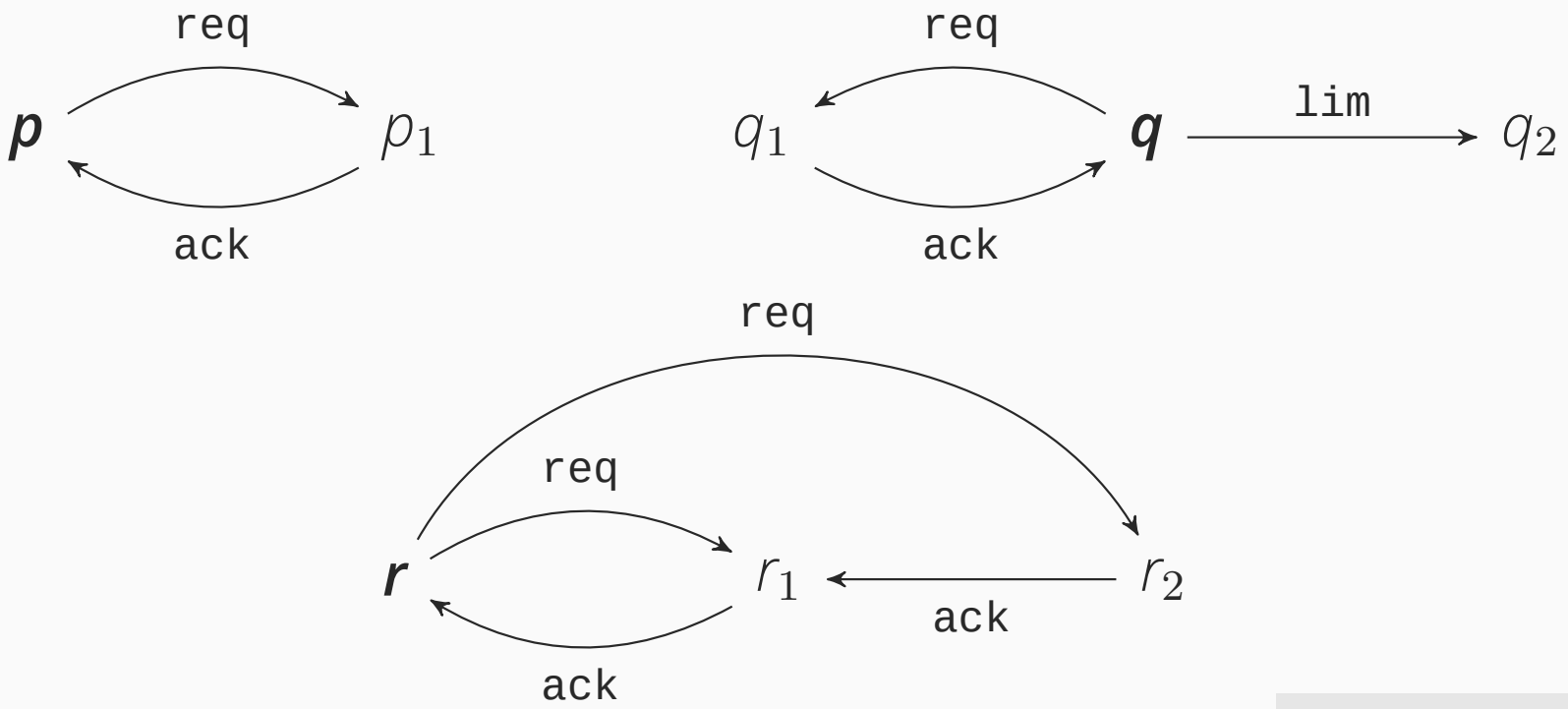
System Properties

$$[\texttt{lim}][\texttt{req}]\mathbf{ff}$$



System Properties

$$p, q, r \in [\![\, [\texttt{lim}][\texttt{req}]\textbf{ff} \,]\!]$$



Definition (Modalities Over action Sets)

Assume Act $= \{\alpha_1, \ldots, \alpha_n\}$

Definition (Modalities Over action Sets)

Assume $\text{Act} = \{\alpha_1, \ldots, \alpha_n\}$

$$[\text{Act}]\varphi \stackrel{\text{def}}{=} [\alpha_1]\varphi \wedge \ldots \wedge [\alpha_n]\varphi$$

Definition (Modalities Over action Sets)

Assume $\text{Act} = \{\alpha_1, \ldots, \alpha_n\}$

$$[\text{Act}]\varphi \stackrel{\text{def}}{=} [\alpha_1]\varphi \wedge \ldots \wedge [\alpha_n]\varphi \qquad \langle\text{Act}\rangle\varphi \stackrel{\text{def}}{=} \langle\alpha_1\rangle\varphi \vee \ldots \vee \langle\alpha_n\rangle\varphi$$

Definition (Modalities Over action Sets)

Assume $\text{Act} = \{\alpha_1, \ldots, \alpha_n\}$

$$[\text{Act}]\varphi \stackrel{\text{def}}{=} [\alpha_1]\varphi \wedge \ldots \wedge [\alpha_n]\varphi \qquad \langle\text{Act}\rangle\varphi \stackrel{\text{def}}{=} \langle\alpha_1\rangle\varphi \vee \ldots \vee \langle\alpha_n\rangle\varphi$$

Invariance and Possibility over Act

$$\mathbf{inv}(\varphi)$$

Definition (Modalities Over action Sets)

Assume $\text{Act} = \{\alpha_1, \ldots, \alpha_n\}$

$$[\text{Act}]\varphi \stackrel{\text{def}}{=} [\alpha_1]\varphi \wedge \ldots \wedge [\alpha_n]\varphi \qquad \langle\text{Act}\rangle\varphi \stackrel{\text{def}}{=} \langle\alpha_1\rangle\varphi \vee \ldots \vee \langle\alpha_n\rangle\varphi$$

Invariance and Possibility over Act

$$\textbf{inv}(\varphi) \stackrel{\text{def}}{=} \varphi \wedge [\text{Act}]\varphi \wedge [\text{Act}][\text{Act}]\varphi \wedge \ldots$$

Definition (Modalities Over action Sets)

Assume $\text{Act} = \{\alpha_1, \ldots, \alpha_n\}$

$$[\text{Act}]\varphi \stackrel{\text{def}}{=} [\alpha_1]\varphi \wedge \ldots \wedge [\alpha_n]\varphi \qquad \langle\text{Act}\rangle\varphi \stackrel{\text{def}}{=} \langle\alpha_1\rangle\varphi \vee \ldots \vee \langle\alpha_n\rangle\varphi$$

Invariance and Possibility over Act

$$\textbf{inv}(\varphi) \stackrel{\text{def}}{=} \varphi \wedge [\text{Act}]\varphi \wedge [\text{Act}][\text{Act}]\varphi \wedge \ldots$$

$$\textbf{pos}(\varphi)$$

The Logic

Definition (Modalities Over action Sets)

Assume $\text{Act} = \{\alpha_1, \ldots, \alpha_n\}$

$$[\text{Act}]\varphi \stackrel{\text{def}}{=} [\alpha_1]\varphi \wedge \ldots \wedge [\alpha_n]\varphi \qquad \langle\text{Act}\rangle\varphi \stackrel{\text{def}}{=} \langle\alpha_1\rangle\varphi \vee \ldots \vee \langle\alpha_n\rangle\varphi$$

Invariance and Possibility over Act

$$\textbf{inv}(\varphi) \stackrel{\text{def}}{=} \varphi \wedge [\text{Act}]\varphi \wedge [\text{Act}][\text{Act}]\varphi \wedge \ldots$$

$$\textbf{pos}(\varphi) \stackrel{\text{def}}{=} \varphi \vee \langle\text{Act}\rangle\varphi \vee \langle\text{Act}\rangle\langle\text{Act}\rangle\varphi \vee \ldots$$

The Logic

Definition (Modalities Over action Sets)

Assume $\text{Act} = \{\alpha_1, \ldots, \alpha_n\}$

$$[\text{Act}]\varphi \stackrel{\text{def}}{=} [\alpha_1]\varphi \wedge \ldots \wedge [\alpha_n]\varphi \qquad \langle\text{Act}\rangle\varphi \stackrel{\text{def}}{=} \langle\alpha_1\rangle\varphi \vee \ldots \vee \langle\alpha_n\rangle\varphi$$

Invariance and Possibility over Act

$$\textbf{inv}(\varphi) \stackrel{\text{def}}{=} (X = \varphi \wedge [\text{Act}]X)$$

$$\textbf{pos}(\varphi) \stackrel{\text{def}}{=} \varphi \vee \langle\text{Act}\rangle\varphi \vee \langle\text{Act}\rangle\langle\text{Act}\rangle\varphi \vee \ldots$$

Definition (Modalities Over action Sets)

Assume $\text{Act} = \{\alpha_1, \ldots, \alpha_n\}$

$$[\text{Act}]\varphi \stackrel{\text{def}}{=} [\alpha_1]\varphi \wedge \ldots \wedge [\alpha_n]\varphi \qquad \langle\text{Act}\rangle\varphi \stackrel{\text{def}}{=} \langle\alpha_1\rangle\varphi \vee \ldots \vee \langle\alpha_n\rangle\varphi$$

Invariance and Possibility over Act

$$\textbf{inv}(\varphi) \stackrel{\text{def}}{=} (X = \varphi \wedge [\text{Act}]X)$$

$$\textbf{pos}(\varphi) \stackrel{\text{def}}{=} (X = \varphi \vee \langle\text{Act}\rangle X)$$

Quiz

Fix Points in a tiny Nutshell

Solving recursive equations

- Consider the following equations over *Nat*:

Fix Points in a tiny Nutshell

Solving recursive equations

- Consider the following equations over *Nat*:
  - $x = 2 \times x$ has **one** solution $(x = 0)$

Fix Points in a tiny Nutshell

Solving recursive equations

- Consider the following equations over *Nat*:
  - $x = 2 \times x$ has **one** solution ($x = 0$)
  - $x = 1 \times x$ has **many** solutions (all $x = n \in Nat$)

Fix Points in a tiny Nutshell

Solving recursive equations

- Consider the following equations over *Nat*:
  - $x = 2 \times x$ has **one** solution ($x = 0$)
  - $x = 1 \times x$ has **many** solutions (all $x = n \in Nat$)
  - $x = 1 + x$ has **???**

Fix Points in a tiny Nutshell

Solving recursive equations

- Consider the following equations over *Nat*:
  - $x = 2 \times x$ has **one** solution ($x = 0$)
  - $x = 1 \times x$ has **many** solutions (all $x = n \in Nat$)
  - $x = 1 + x$ has **no** solutions!

FixPoints in a tiny Nutshell

Solving recursive equations

- Consider the following equations over *Nat*:
  - $x = 2 \times x$ has **one** solution ($x = 0$)
  - $x = 1 \times x$ has **many** solutions (all $x = n \in Nat$)
  - $x = 1 + x$ has **no** solutions!
- Equations over 2^{Nat} (sets of integers)

Fix Points in a tiny Nutshell

Solving recursive equations

- Consider the following equations over *Nat*:
  - $x = 2 \times x$ has **one** solution ($x = 0$)
  - $x = 1 \times x$ has **many** solutions (all $x = n \in Nat$)
  - $x = 1 + x$ has **no** solutions!
- Equations over 2^{Nat} (sets of integers)
  - $x = (\{1\} \cup x) \cap \{1\}$ has **one** solution ($x = \{1\}$)

Fix Points in a tiny Nutshell

Solving recursive equations

- Consider the following equations over *Nat*:
  - $x = 2 \times x$ has **one** solution ($x = 0$)
  - $x = 1 \times x$ has **many** solutions (all $x = n \in Nat$)
  - $x = 1 + x$ has **no** solutions!
- Equations over 2^{Nat} (sets of integers)
  - $x = (\{1\} \cup x) \cap \{1\}$ has **one** solution ($x = \{1\}$)
  - $x = \{1\} \cap x$ has **two** solutions (**???**)

Fix Points in a tiny Nutshell

Solving recursive equations

- Consider the following equations over *Nat*:
  - $x = 2 \times x$ has **one** solution ($x = 0$)
  - $x = 1 \times x$ has **many** solutions (all $x = n \in Nat$)
  - $x = 1 + x$ has **no** solutions!
- Equations over 2^{Nat} (sets of integers)
  - $x = (\{1\} \cup x) \cap \{1\}$ has **one** solution ($x = \{1\}$)
  - $x = \{1\} \cap x$ has **two** solutions ($x = \{1\}$ and $x = \emptyset$)

FixPoints in a tiny Nutshell

Solving recursive equations

- Consider the following equations over *Nat*:
  - $x = 2 \times x$ has **one** solution ($x = 0$)
  - $x = 1 \times x$ has **many** solutions (all $x = n \in Nat$)
  - $x = 1 + x$ has **no** solutions!
- Equations over 2^{Nat} (sets of integers)
  - $x = (\{1\} \cup x) \cap \{1\}$ has **one** solution ($x = \{1\}$)
  - $x = \{1\} \cap x$ has **two** solutions ($x = \{1\}$ and $x = \emptyset$)
  - $x = \{1\} \cup x$ has **many** solutions (all N where $\{1\} \subseteq N$)

Fix Points in a tiny Nutshell

Solving recursive equations

- Consider the following equations over *Nat*:
  - $x = 2 \times x$ has **one** solution ($x = 0$)
  - $x = 1 \times x$ has **many** solutions (all $x = n \in Nat$)
  - $x = 1 + x$ has **no** solutions!
- Equations over 2^{Nat} (sets of integers)
  - $x = (\{1\} \cup x) \cap \{1\}$ has **one** solution ($x = \{1\}$)
  - $x = \{1\} \cap x$ has **two** solutions ($x = \{1\}$ and $x = \emptyset$)
  - $x = \{1\} \cup x$ has **many** solutions (all N where $\{1\} \subseteq N$)
  - $x = Nat \setminus x$ has **no** solutions!

Fix Points in a tiny Nutshell

FixPoints in a tiny Nutshell

- Formulas over Proc with one free variable can be seen as a **(monotonic) function** $2^{\text{Sys}} \mapsto 2^{\text{Sys}}$.

Fix Points in a tiny Nutshell

- Formulas over Proc with one free variable can be seen as a **(monotonic) function** $2^{\text{Sys}} \mapsto 2^{\text{Sys}}$.

For example $[\alpha]X$

Fix Points in a tiny Nutshell

- Formulas over Proc with one free variable can be seen as a **(monotonic) function** $2^{\text{Sys}} \mapsto 2^{\text{Sys}}$.
- $(2^{\text{Sys}}, \subseteq)$ forms a **complete lattice**.

FixPoints in a tiny Nutshell

- Formulas over Proc with one free variable can be seen as a **(monotonic) function** $2^{\text{Sys}} \mapsto 2^{\text{Sys}}$.
- $(2^{\text{Sys}}, \subseteq)$ forms a **complete lattice**.

Theorem (Tarski's Fixed-Point Theorem)

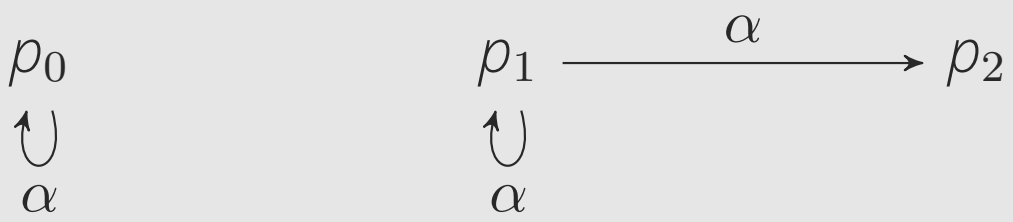
*Let $(D, \subseteq)$ be a complete lattice and $f :: D \mapsto D$ be a monotonic function. Then f has a **unique largest** fixpoint and a **unique smallest** fixpoints, given by:*

$$max \stackrel{def}{=} \bigcup \{x \in D \mid x \subseteq f(x)\}$$

$$min \stackrel{def}{=} \bigcap \{x \in D \mid f(x) \subseteq x\}$$

Simple Sample LTS

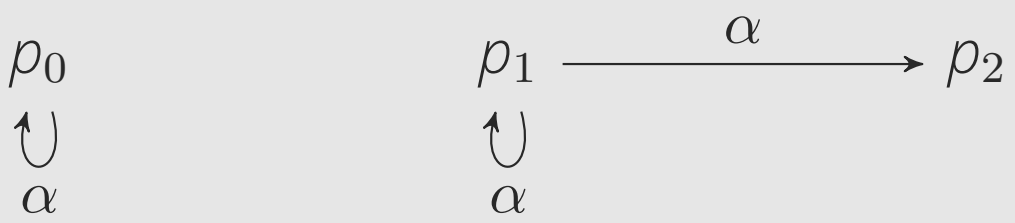
$$\left\langle \{p_0, p_1, p_2\},\ \{\alpha\},\ \left\{ p_0 \xrightarrow{\alpha} p_0, p_1 \xrightarrow{\alpha} p_1, p_1 \xrightarrow{\alpha} p_2 \right\} \right\rangle$$



Largest or Least?

Simple Sample LTS

$$\left\langle \{p_0, p_1, p_2\},\ \{\alpha\},\ \left\{ p_0 \xrightarrow{\alpha} p_0, p_1 \xrightarrow{\alpha} p_1, p_1 \xrightarrow{\alpha} p_2 \right\} \right\rangle$$

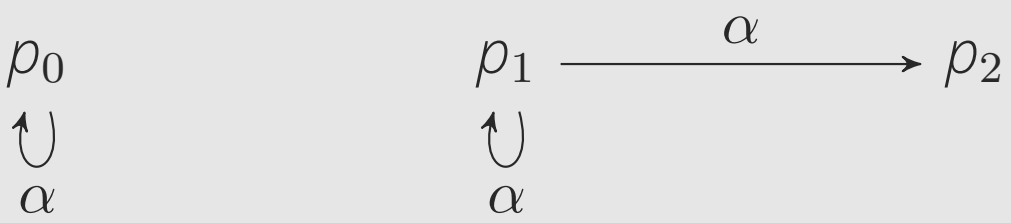


Largest or Least?

inv(φ) expressed as $\quad X = \varphi \wedge [\mathrm{Act}]X$

Simple Sample LTS

$$\left\langle \{p_0, p_1, p_2\},\ \{\alpha\},\ \left\{ p_0 \xrightarrow{\alpha} p_0, p_1 \xrightarrow{\alpha} p_1, p_1 \xrightarrow{\alpha} p_2 \right\} \right\rangle$$

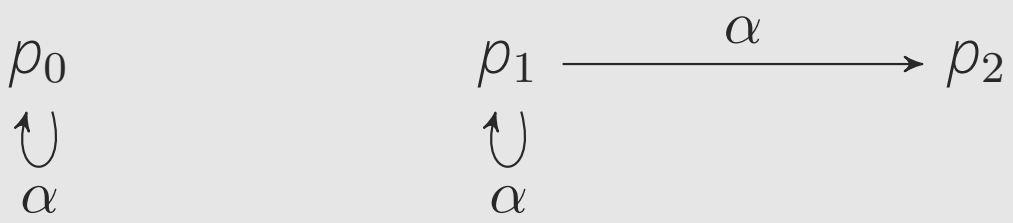


Largest or Least?

inv($\langle\alpha\rangle$**tt**) where Act $= \{\alpha\}$ expressed as $X = \langle\alpha\rangle\mathbf{tt} \wedge [\alpha]X$

Simple Sample LTS

$$\left\langle \{p_0, p_1, p_2\},\ \{\alpha\},\ \left\{ p_0 \xrightarrow{\alpha} p_0, p_1 \xrightarrow{\alpha} p_1, p_1 \xrightarrow{\alpha} p_2 \right\} \right\rangle$$



Largest or Least?

inv($\langle\alpha\rangle$**tt**) where Act $= \{\alpha\}$ expressed as $X = \langle\alpha\rangle\mathbf{tt} \wedge [\alpha]X$

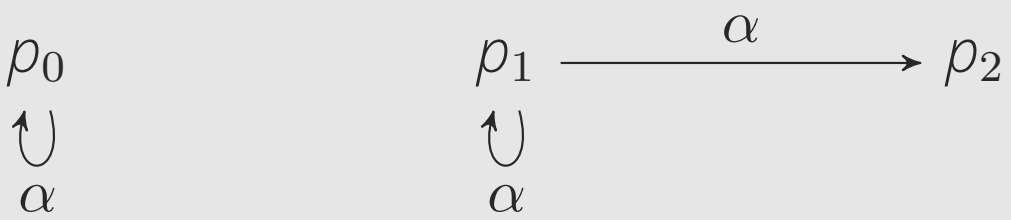
There are two solutions for X:

$$\emptyset \quad \text{or} \quad \{p_0\}$$

We want the **largest** one.

Simple Sample LTS

$$\left\langle \{p_0, p_1, p_2\},\ \{\alpha\},\ \left\{ p_0 \xrightarrow{\alpha} p_0, p_1 \xrightarrow{\alpha} p_1, p_1 \xrightarrow{\alpha} p_2 \right\} \right\rangle$$

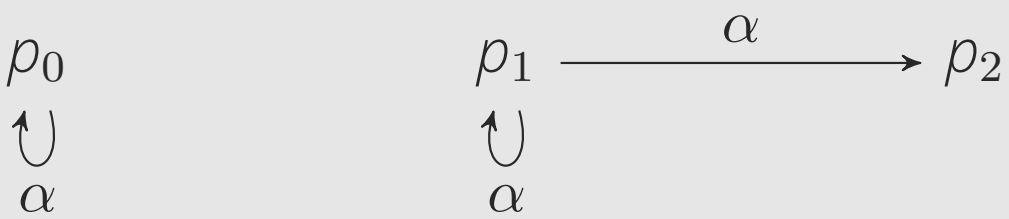


Largest or Least?

pos(φ) expressed as $\quad X = \varphi \ \vee \ \langle \mathrm{Act} \rangle X$

Simple Sample LTS

$$\left\langle \{p_0, p_1, p_2\},\ \{\alpha\},\ \left\{ p_0 \xrightarrow{\alpha} p_0, p_1 \xrightarrow{\alpha} p_1, p_1 \xrightarrow{\alpha} p_2 \right\} \right\rangle$$

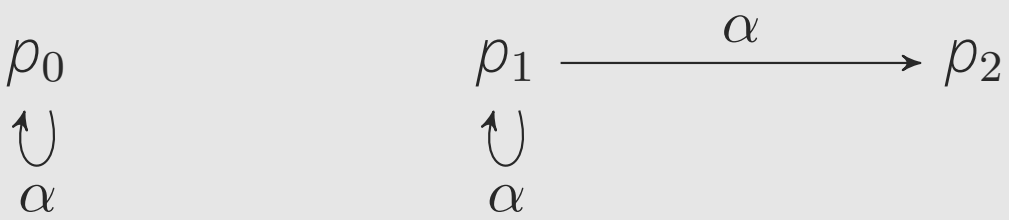


Largest or Least?

pos($[\alpha]$**ff**) expressed as $\quad X = [\alpha]\mathbf{ff} \ \vee \ \langle\alpha\rangle X$

Simple Sample LTS

$$\left\langle \{p_0, p_1, p_2\},\ \{\alpha\},\ \left\{ p_0 \xrightarrow{\alpha} p_0, p_1 \xrightarrow{\alpha} p_1, p_1 \xrightarrow{\alpha} p_2 \right\} \right\rangle$$



Largest or Least?

pos($[\alpha]$**ff**) expressed as $\quad X = [\alpha]\mathbf{ff} \ \vee \ \langle\alpha\rangle X$

There are two solutions for X:

$$\{p_1, p_2\} \quad \text{or} \quad \{p_0, p_1, p_2\}$$

We want the **least** one.

The Full Logic

Definition (Logic Syntax)

$$
\begin{array}{rlll}
\varphi, \phi \in \mu\text{HML} ::= & \mathbf{ff} & & \text{(falsity)} \\
| & \mathbf{tt} & & \text{(truth)} \\
| & \varphi \wedge \phi & & \text{(conjunction)} \\
| & \varphi \vee \phi & & \text{(disjunction)} \\
| & [\alpha]\varphi & & \text{(necessity)} \\
| & \langle\alpha\rangle\varphi & & \text{(possibility)} \\
| & \mathbf{max}X.\varphi & & \textbf{(maximal fixpoint)} \\
| & \mathbf{min}X.\varphi & & \textbf{(minimum fixpoint)} \\
| & X & & \textbf{(recursion variable)}
\end{array}
$$

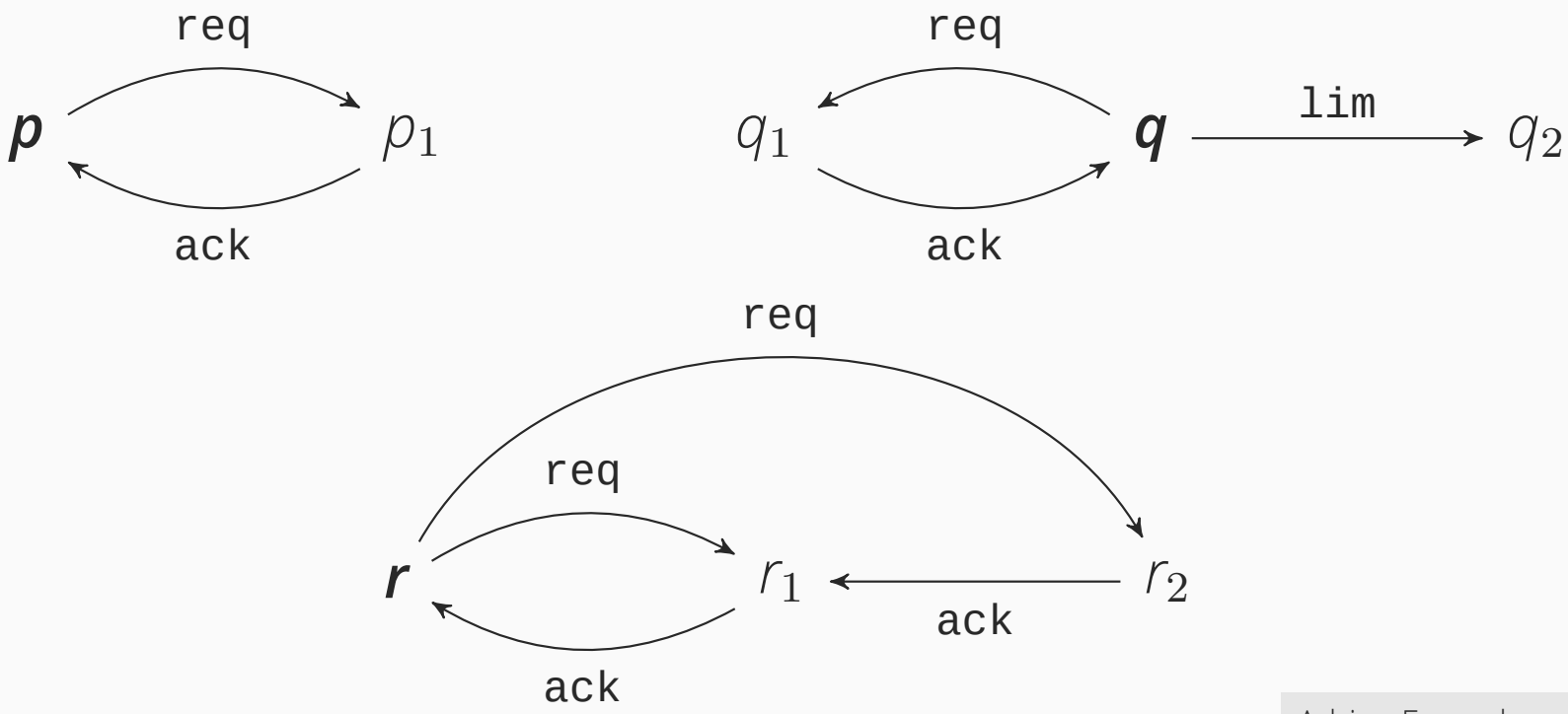
The Logic

Definition (Logic Semantics)

$$
\begin{aligned}
[\![\mathbf{ff}, \rho]\!] &\stackrel{\text{def}}{=} \emptyset \\
[\![\mathbf{tt}, \rho]\!] &\stackrel{\text{def}}{=} \mathrm{Sys} \\
[\![\varphi \wedge \phi, \rho]\!] &\stackrel{\text{def}}{=} [\![\varphi, \rho]\!] \cap [\![\phi, \rho]\!] \\
[\![\varphi \vee \phi, \rho]\!] &\stackrel{\text{def}}{=} [\![\varphi, \rho]\!] \cup [\![\phi, \rho]\!] \\
[\![[\alpha]\varphi, \rho]\!] &\stackrel{\text{def}}{=} \left\{ p \mid \forall p'.\ p \xrightarrow{\alpha} p' \text{ implies } p' \in [\![\varphi, \rho]\!] \right\} \\
[\![\langle\alpha\rangle\varphi, \rho]\!] &\stackrel{\text{def}}{=} \left\{ p \mid \exists p'.\ p \xrightarrow{\alpha} p' \text{ and } p' \in [\![\varphi, \rho]\!] \right\} \\
[\![\mathbf{max}\, X.\varphi, \rho]\!] &\stackrel{\text{def}}{=} \bigcup \{ S \mid S \subseteq [\![\varphi, \rho[X \mapsto S]]\!] \} \\
[\![\mathbf{min}\, X.\varphi, \rho]\!] &\stackrel{\text{def}}{=} \bigcap \{ S \mid [\![\varphi, \rho[X \mapsto S]]\!] \subseteq S \} \\
[\![X, \rho]\!] &\stackrel{\text{def}}{=} \rho(X)
\end{aligned}
$$

System Properties

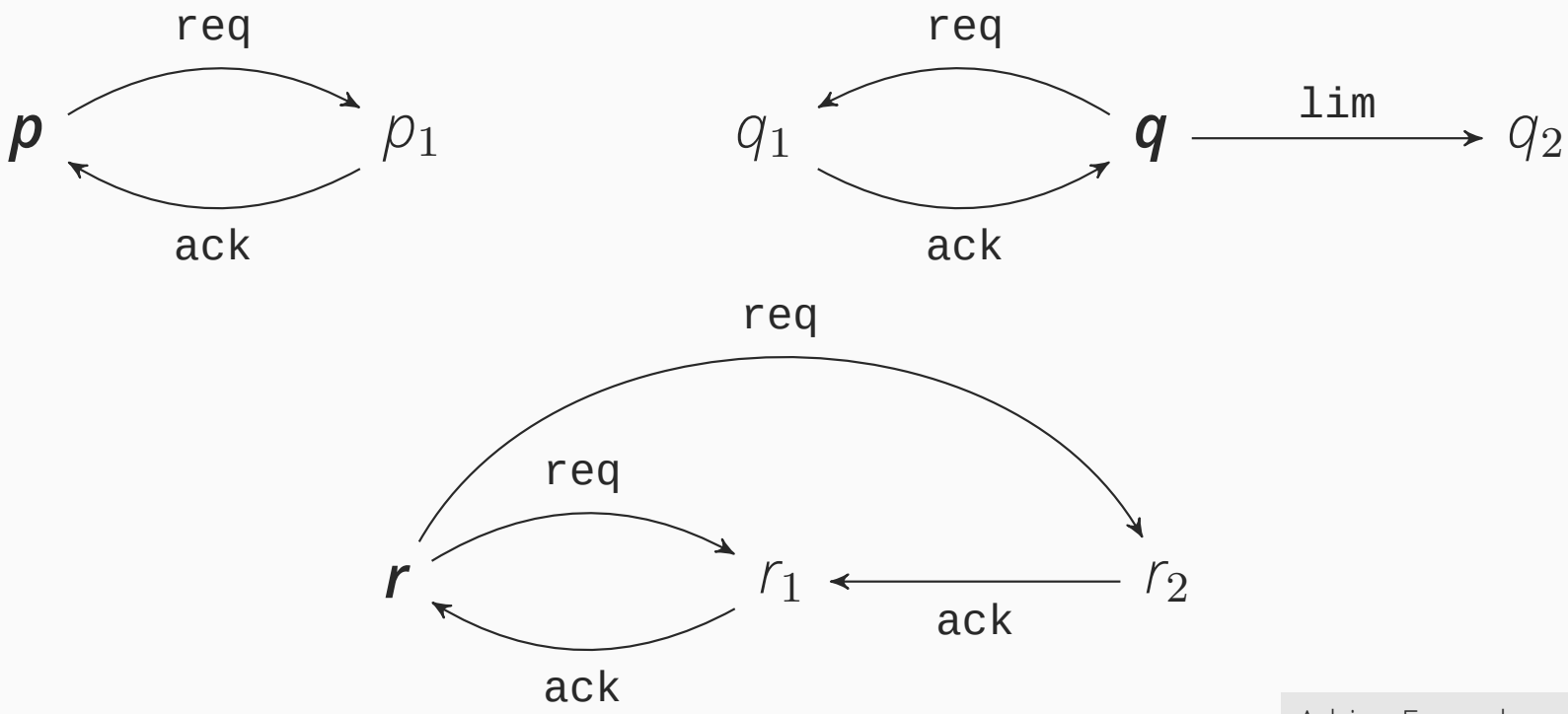
$$\mathbf{max}X.([\mathtt{req}]\,([\mathtt{ack}][\mathtt{ack}]\mathbf{ff} \wedge [\mathtt{ack}]X))$$



The Logic

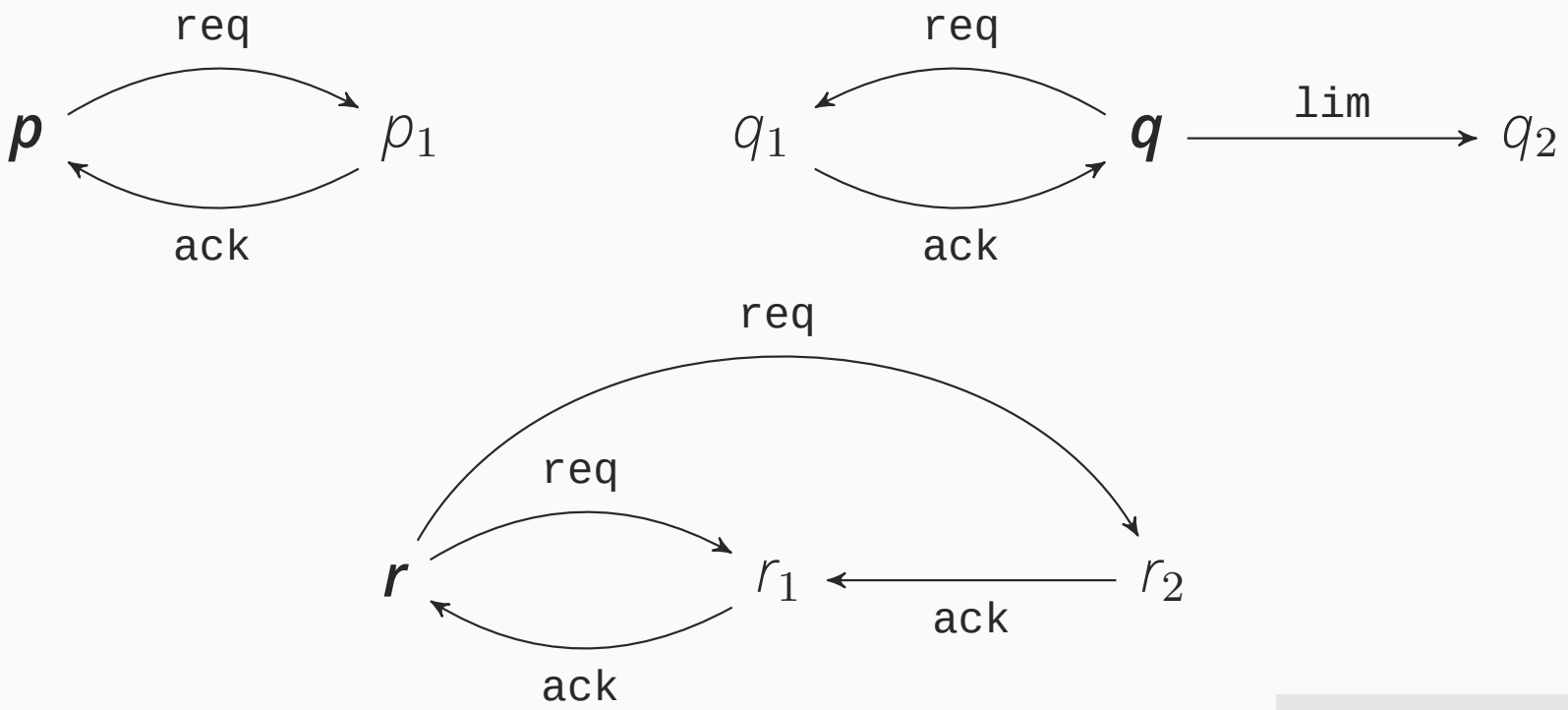
System Properties

$$p, q \in [\![\textbf{max} X.([\texttt{req}]\,([\texttt{ack}][\texttt{ack}]\textbf{ff} \wedge [\texttt{ack}]X))]\!]$$



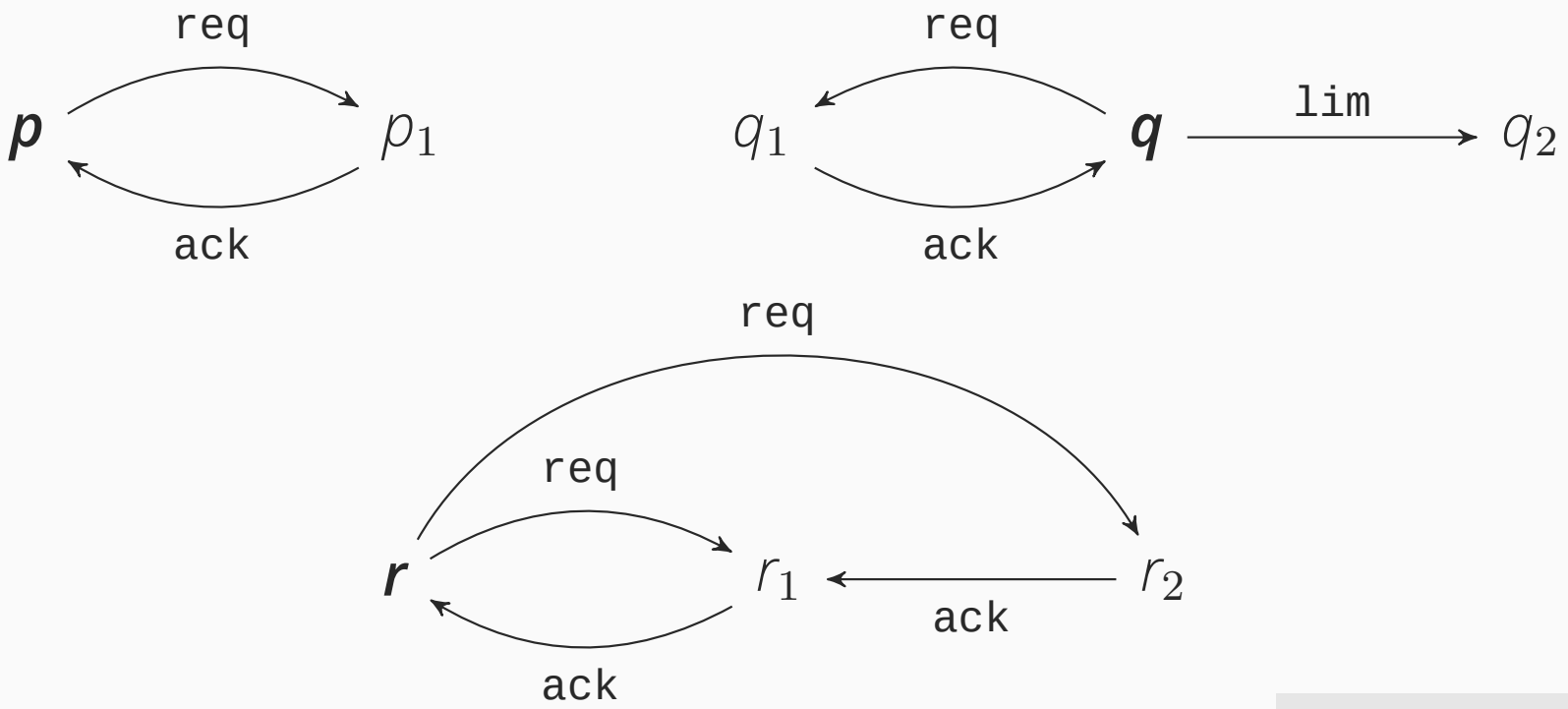
System Properties

$$\mathbf{min}\, X.(\langle \mathtt{lim} \rangle \mathbf{tt} \vee \langle \mathtt{req} \rangle \langle \mathtt{ack} \rangle X)$$



System Properties

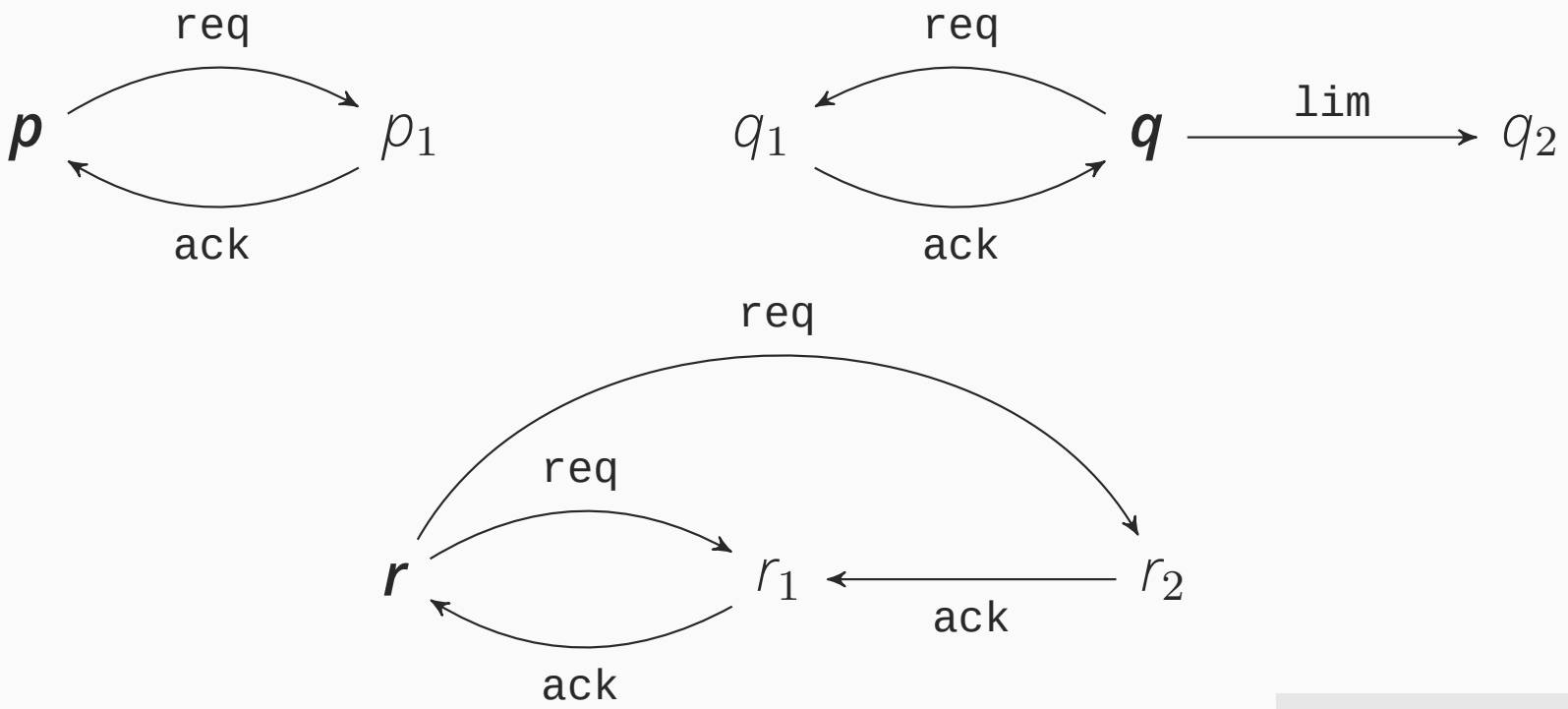
$$q \in [\![\mathbf{min} X.(\langle \mathtt{lim} \rangle \mathbf{tt} \vee \langle \mathtt{req} \rangle \langle \mathtt{ack} \rangle X)]\!]$$



Quiz

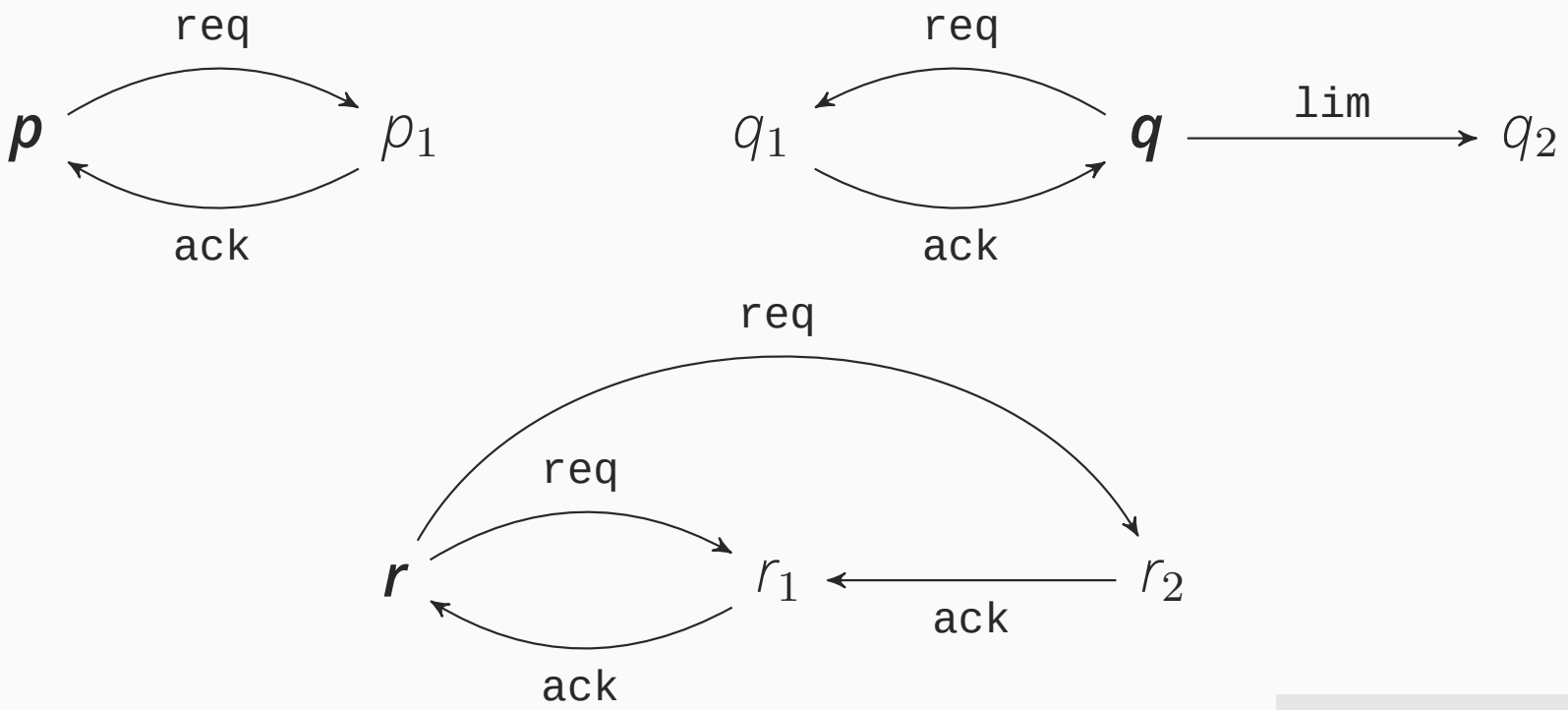
System Properties

$$q \in [\![\textbf{min} X.(\langle \texttt{lim} \rangle \textbf{tt} \vee \langle \texttt{req} \rangle \langle \texttt{ack} \rangle X)]\!]$$



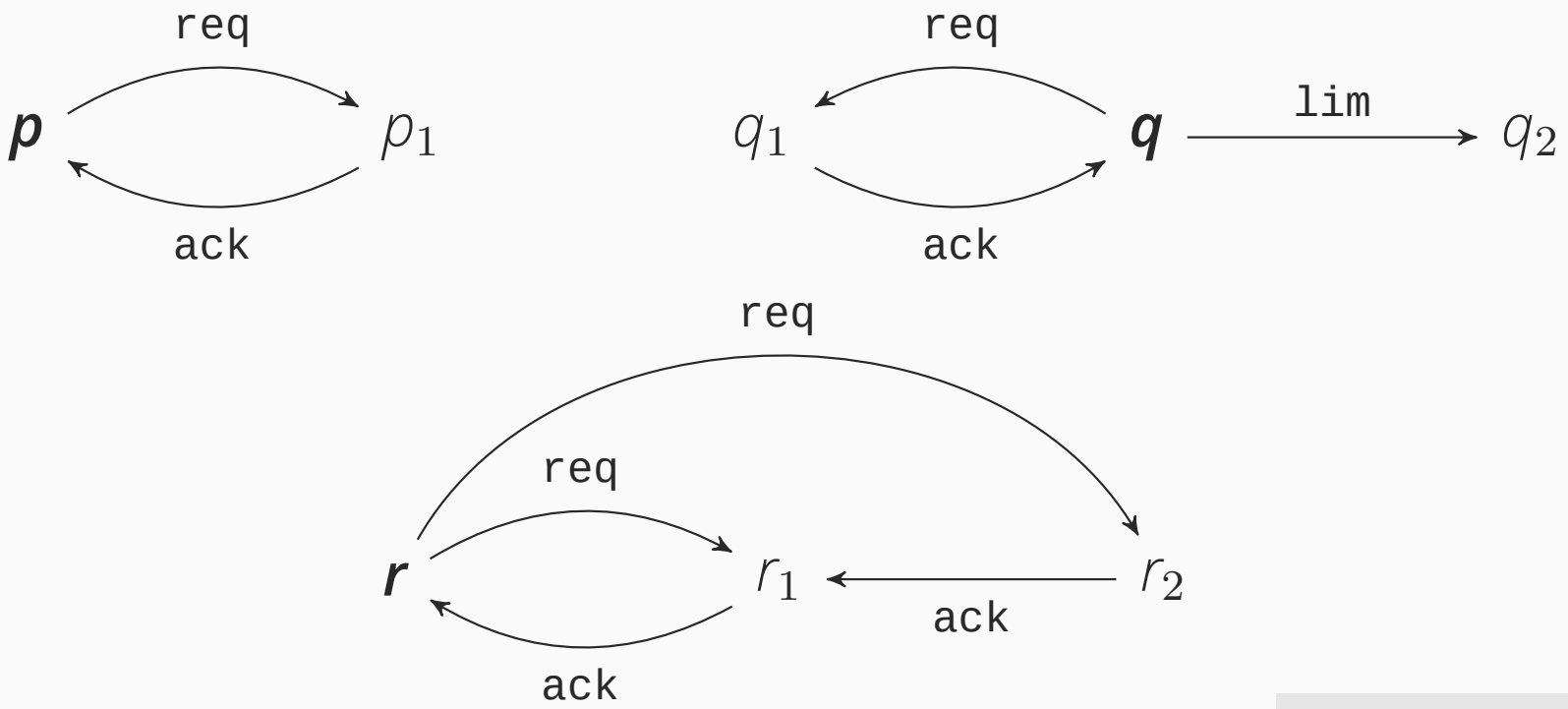
System Properties

$$??? \in [\![\mathbf{max} X.(\langle \texttt{lim} \rangle \mathbf{tt} \vee \langle \texttt{req} \rangle \langle \texttt{ack} \rangle X)]\!]$$



System Properties

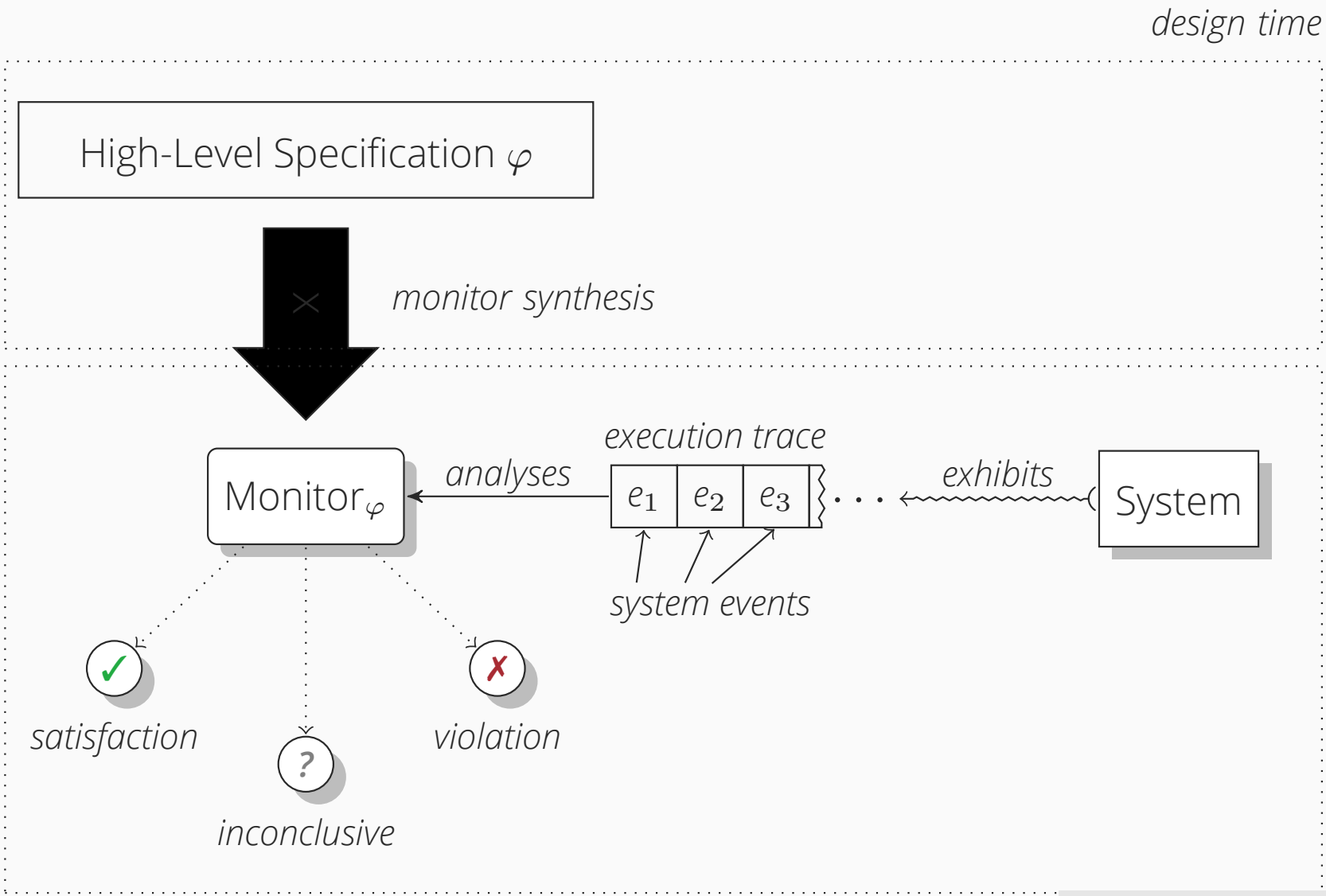
$$p, q, r \in [\![\, \mathbf{max} X.(\langle \mathtt{lim} \rangle \mathbf{tt} \vee \langle \mathtt{req} \rangle \langle \mathtt{ack} \rangle X) \,]\!]$$



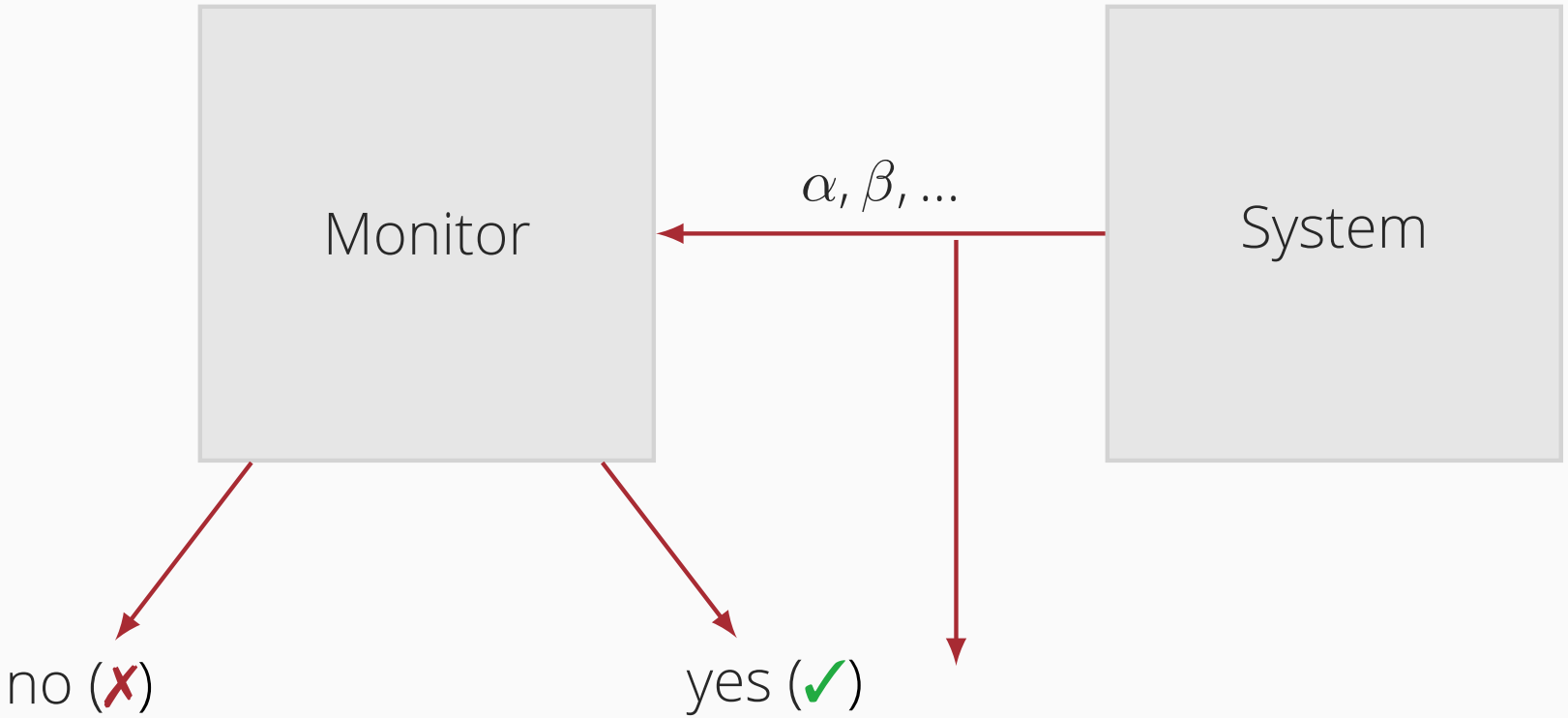
Monitors and Instrumentation

Adrian Francalanza ·

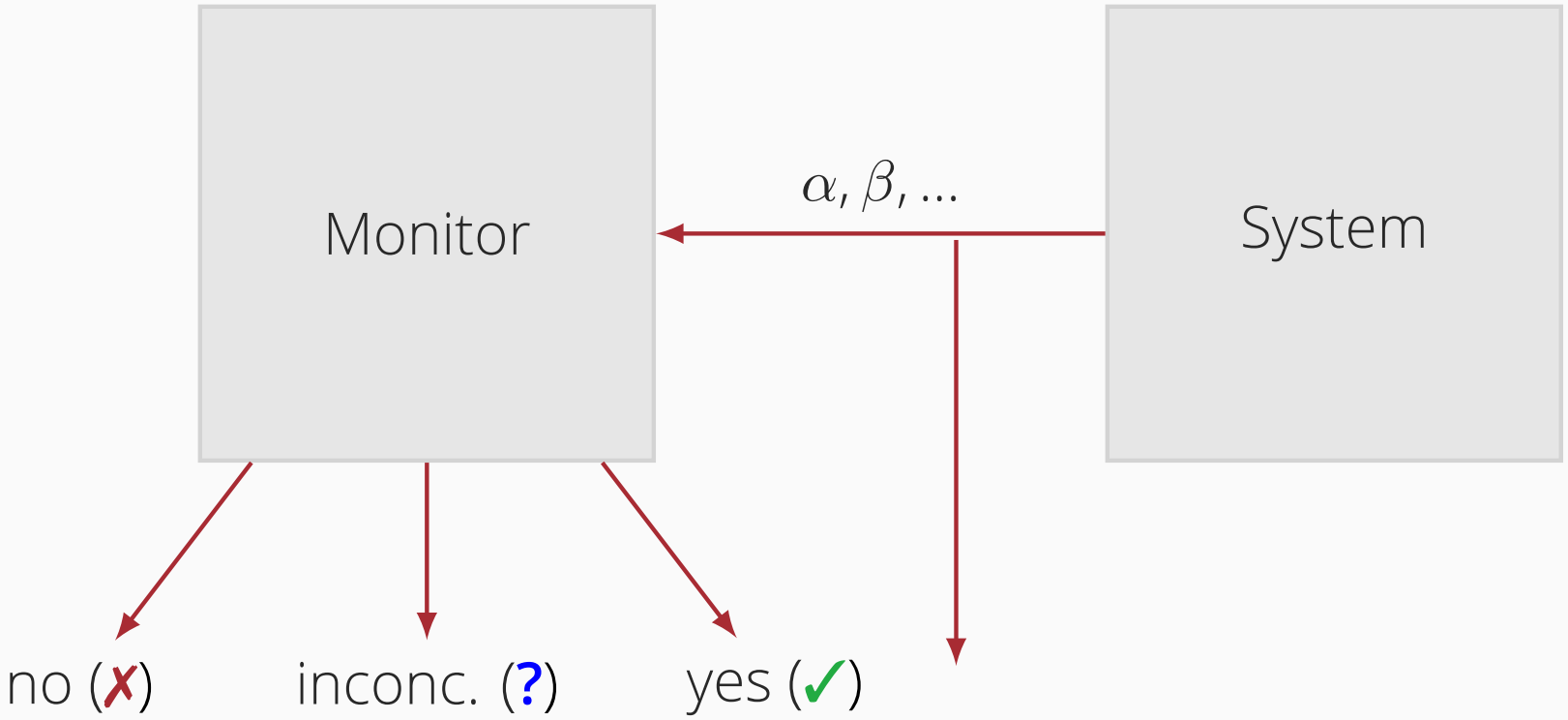
The General Approach



The Basic Setup



The Basic Setup



Monitors

Definition (Monitor Syntax)

$$
\begin{array}{rrllll}
m, n \in \text{Mon} & ::= & v & \mid \alpha.m & \mid m + n & \mid \textbf{rec}\, x.m \quad \mid x \\
v, u \in \text{Verd} & ::= & ? & \mid \times & \mid \checkmark &
\end{array}
$$

Monitors

Definition (Monitor Syntax)

$$m, n \in \text{Mon} ::= v \mid \alpha.m \mid m + n \mid \textbf{rec}\, x.m \mid x$$
$$v, u \in \text{Verd} ::= ? \mid ✗ \mid ✓$$

Monitor Descriptions

$$m_1 = (\texttt{req.ack.ack.}✗ + \texttt{lim.}✓)$$

$$m_2 = \textbf{rec}\, x.(\texttt{lim.}✗ + \texttt{req.ack.}x)$$

Monitors

Definition (Monitor Semantics)

$$\text{mAct}\ \frac{}{\alpha.m \xrightarrow{\alpha} m} \qquad \text{mRec}\ \frac{m[\mathbf{rec}\,x.m/x] \xrightarrow{\alpha} n}{\mathbf{rec}\,x.m \xrightarrow{\alpha} n}$$

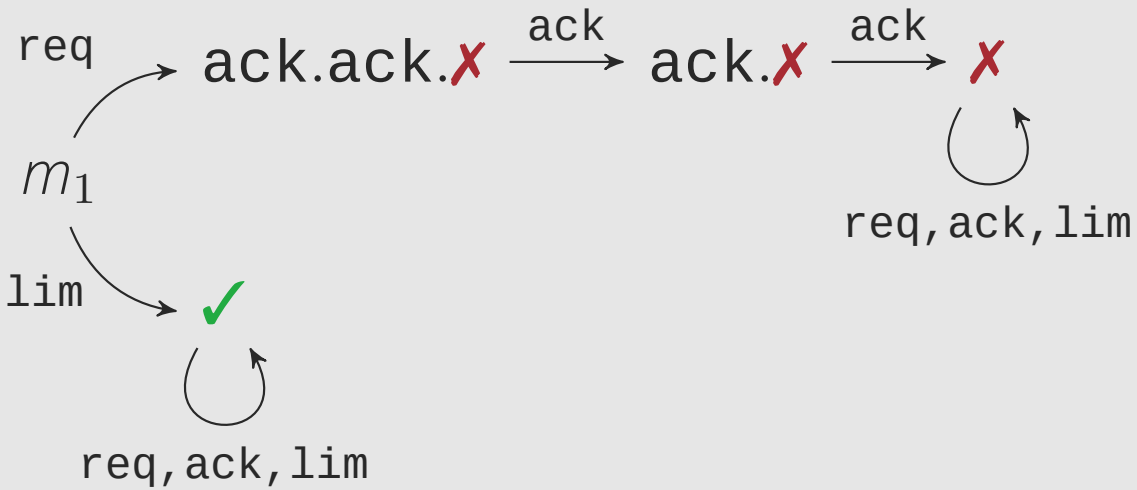
$$\text{mSelL}\ \frac{m \xrightarrow{\alpha} m'}{m + n \xrightarrow{\alpha} m'} \qquad \text{mSelR}\ \frac{n \xrightarrow{\alpha} n'}{m + n \xrightarrow{\alpha} n'}$$

$$\text{mVer}\ \frac{}{v \xrightarrow{\alpha} v}$$

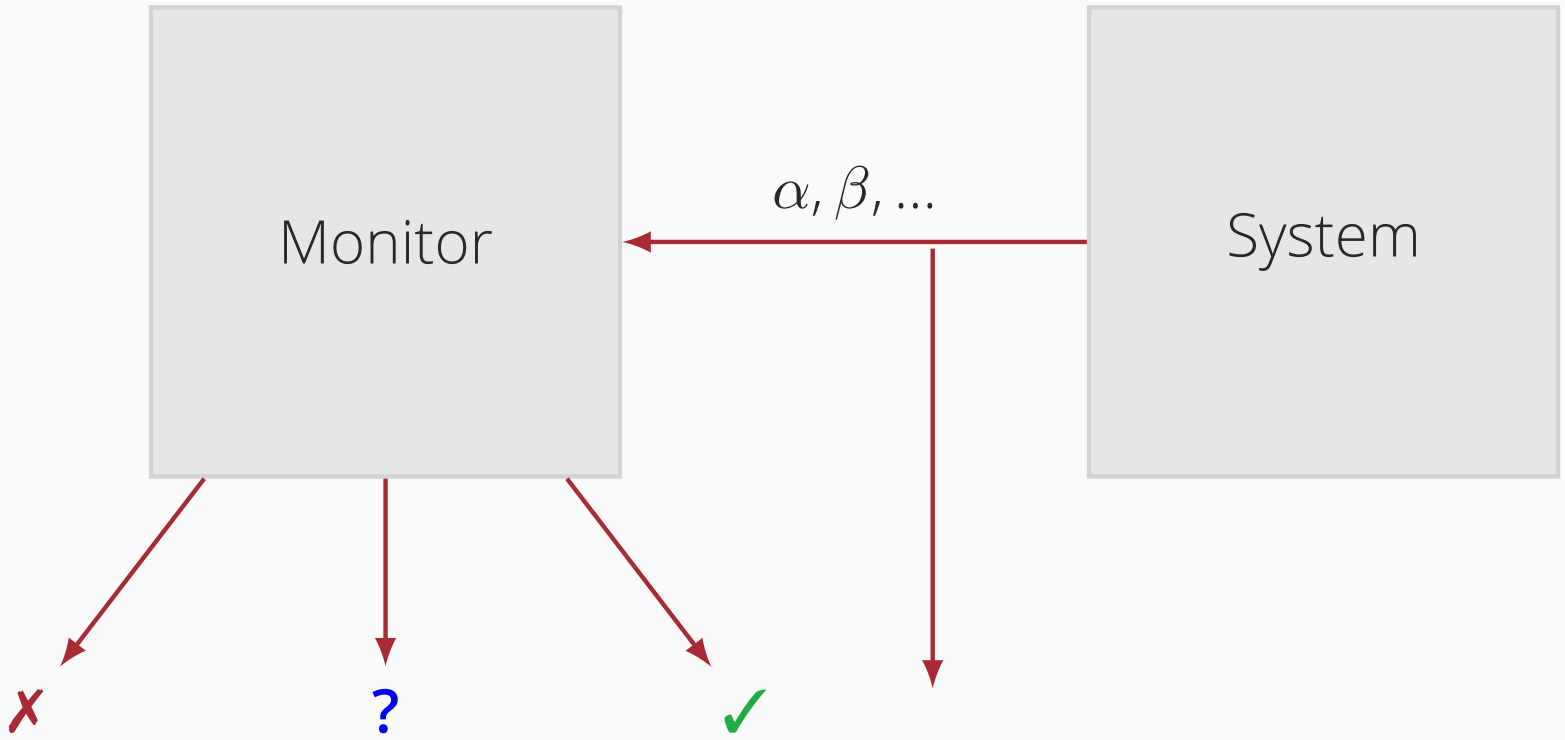
Monitors

From syntax to LTSs

$$m_1 = (\texttt{req.ack.ack.}\times + \texttt{lim.}\checkmark)$$



The Basic Setup



System Instrumentation

Definition (Instrumentation - Simple)

$$\text{Mon}\ \frac{p \xrightarrow{\alpha} p' \qquad m \xrightarrow{\alpha} m'}{m \triangleleft p \xrightarrow{\alpha} m' \triangleleft p'}$$

$$\text{Ter}\ \frac{p \xrightarrow{\alpha} p' \qquad m \not\xrightarrow{\alpha}}{m \triangleleft p \xrightarrow{\alpha} ? \triangleleft p'}$$

System Instrumentation

Definition (Instrumentation - Simple)

$$\text{Mon}\ \frac{p \xrightarrow{\alpha} p' \qquad m \xrightarrow{\alpha} m'}{m \triangleleft p \xrightarrow{\alpha} m' \triangleleft p'}$$

$$\text{Ter}\ \frac{p \xrightarrow{\alpha} p' \qquad m \not\xrightarrow{\alpha}}{m \triangleleft p \xrightarrow{\alpha} ? \triangleleft p'}$$

- The execution **trace** is consumed **incrementally**.
- The monitor is **passive** - the system drives the computation.

System Instrumentation

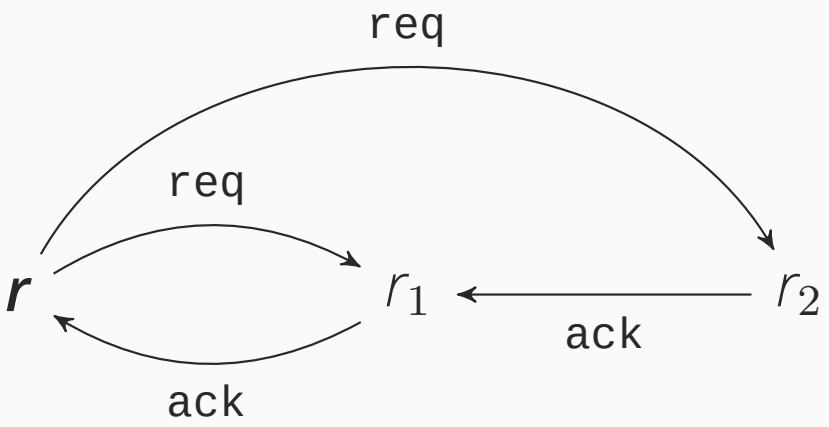
Definition (Instrumentation)

$$\text{Mon}\ \frac{p \xrightarrow{\alpha} p' \qquad m \xrightarrow{\alpha} m'}{m \triangleleft p \xrightarrow{\alpha} m' \triangleleft p'}$$

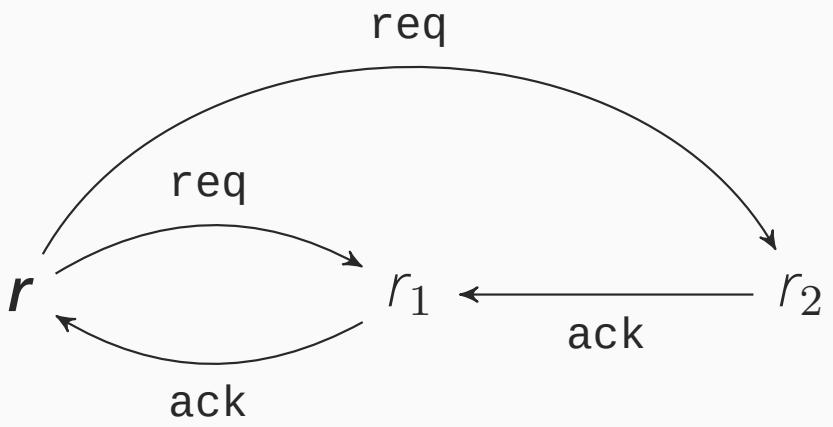
$$\text{Ter}\ \frac{p \xrightarrow{\alpha} p' \qquad m \not\xrightarrow{\alpha} \qquad m \not\xrightarrow{\tau}}{m \triangleleft p \xrightarrow{\alpha} ? \triangleleft p'}$$

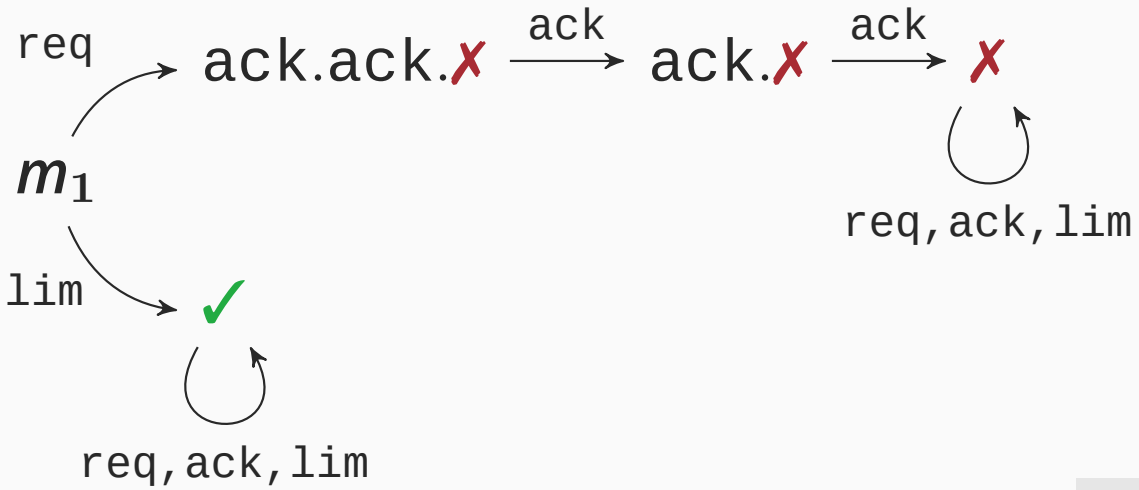
$$\text{AsS}\ \frac{p \xrightarrow{\tau} p'}{m \triangleleft p \xrightarrow{\tau} m \triangleleft p'} \qquad \text{AsM}\ \frac{m \xrightarrow{\tau} m'}{m \triangleleft p \xrightarrow{\tau} m' \triangleleft p}$$

System Instrumentation

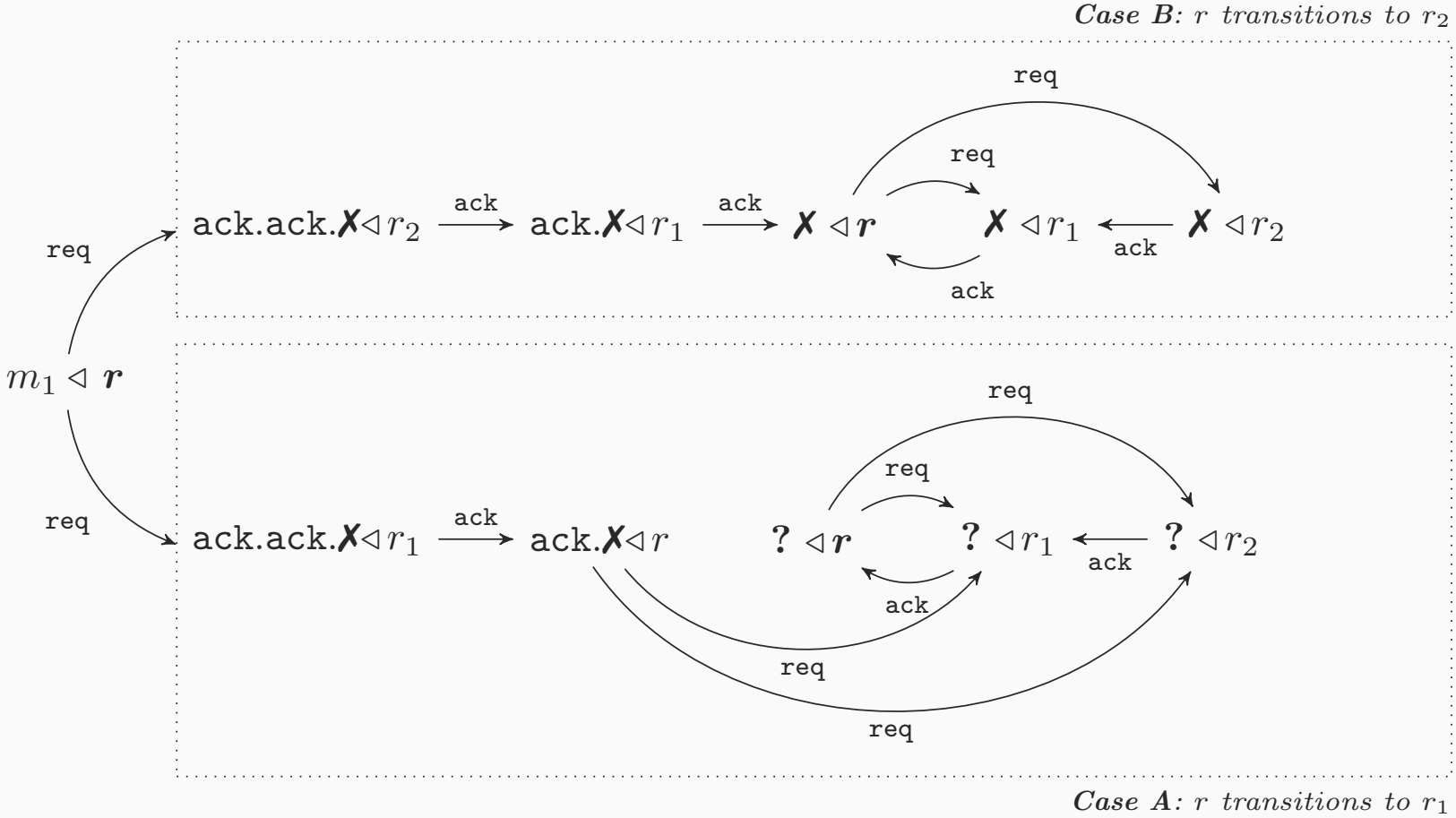


System Instrumentation



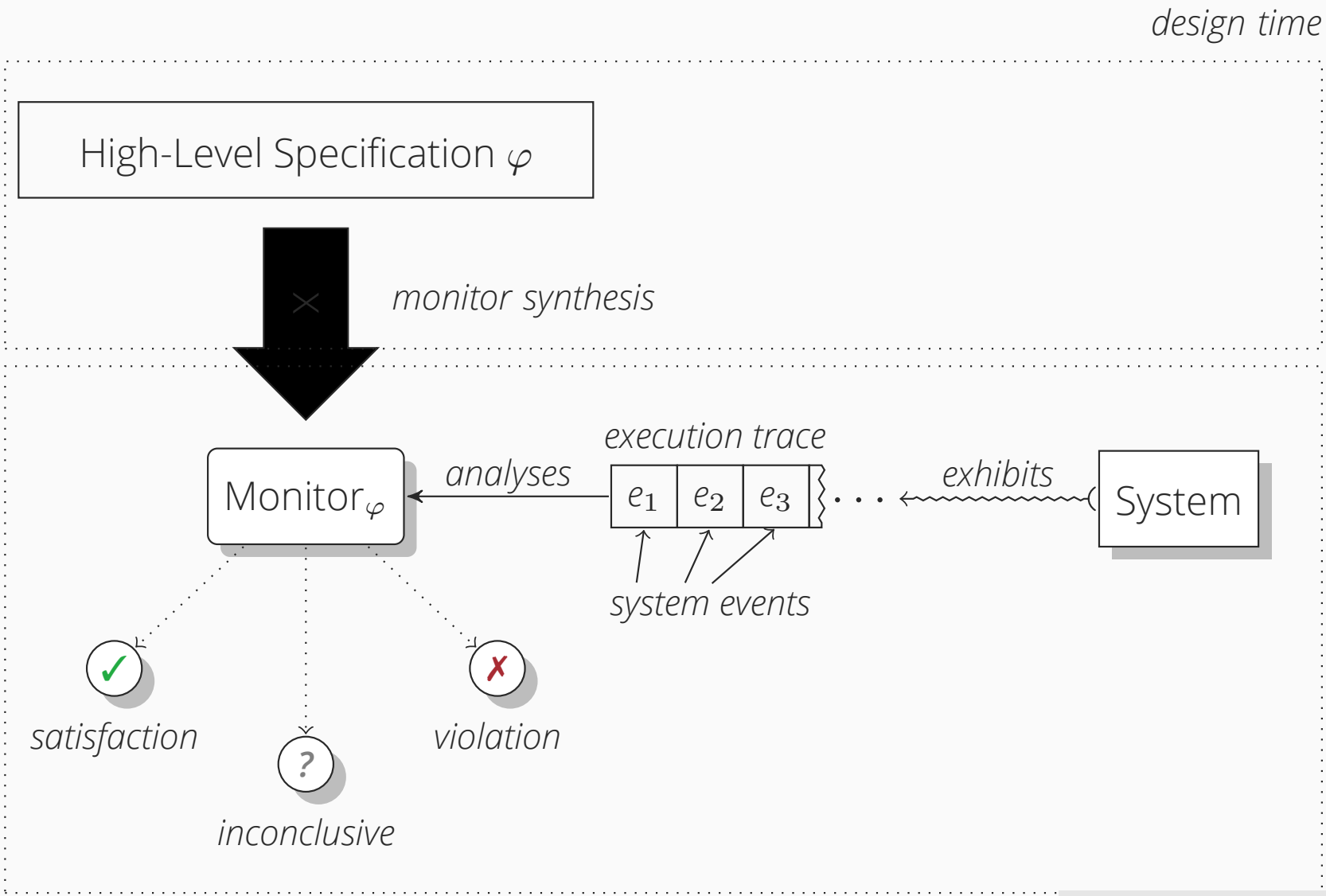


System Instrumentation



Monitorability

The General Approach



Verdicts Vs Satisfactions and Violations

Logic

Monitoring

Verdicts Vs Satisfactions and Violations

Logic

$p \in [\![\varphi]\!]$
$p \notin [\![\varphi]\!]$

Monitoring

Verdicts Vs Satisfactions and Violations

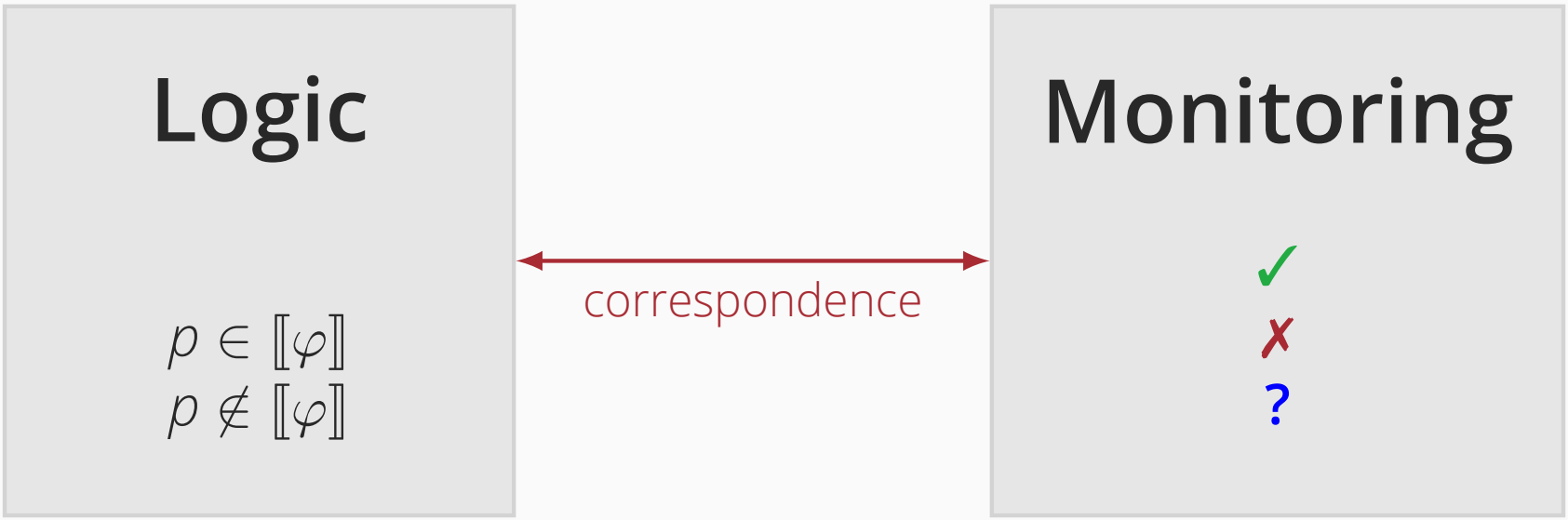
Logic

$p \in [\![\varphi]\!]$
$p \notin [\![\varphi]\!]$

Monitoring

✓
✗
?

Verdicts Vs Satisfactions and Violations



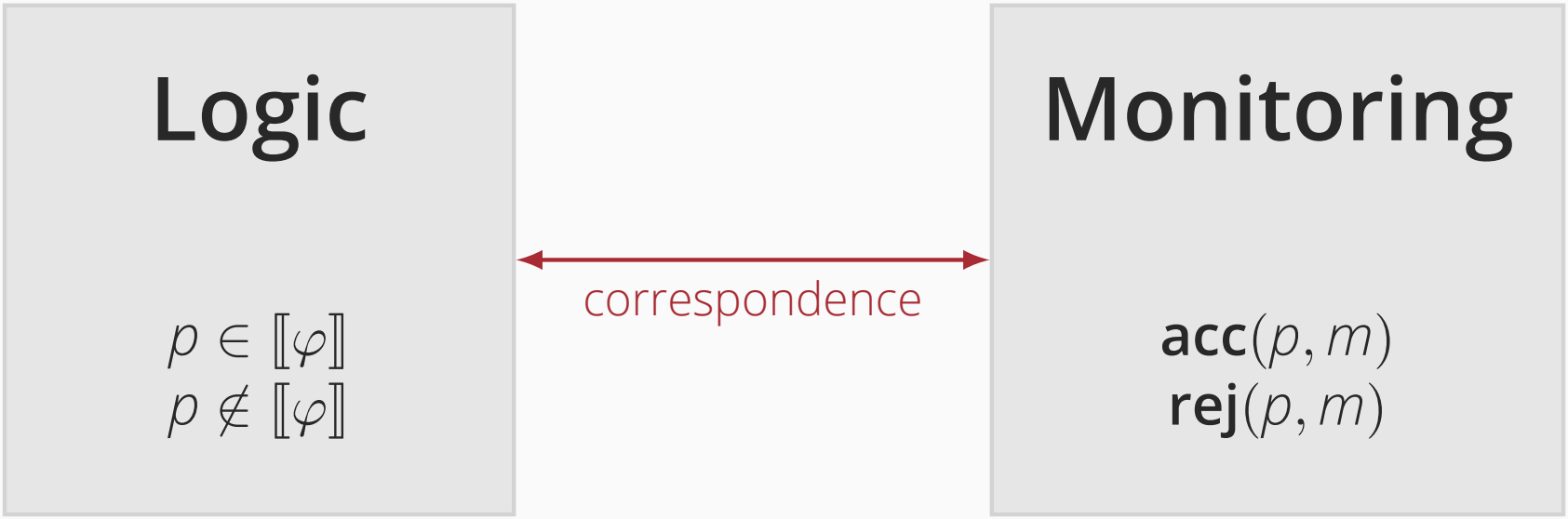
Monitoring Verdicts

Definition (Acceptances and Rejections)

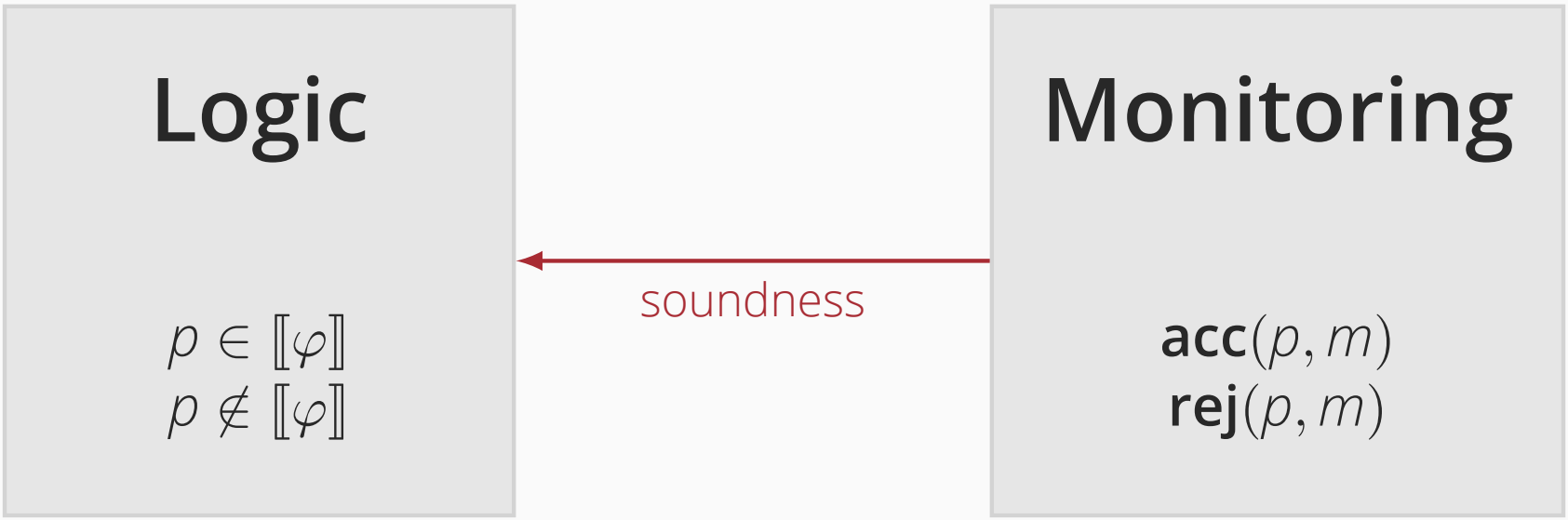
$$\mathbf{acc}(p, m) \stackrel{\text{def}}{=} \exists t, p' \quad \text{such that} \quad m \triangleleft p \stackrel{t}{\Rightarrow} \checkmark \triangleleft p'$$

$$\mathbf{rej}(p, m) \stackrel{\text{def}}{=} \exists t, p' \quad \text{such that} \quad m \triangleleft p \stackrel{t}{\Rightarrow} \times \triangleleft p'$$

Verdicts Vs Satisfactions and Violations



Verdicts Vs Satisfactions and Violations



Soundness

Definition (Sound Monitoring)

Monitor m **soundly** monitors for the property φ, denoted as **smon**(m, φ), whenever for **every** system $p \in \mathrm{Sys}$ we have:

- **acc**(p, m) implies $p \in [\![\varphi]\!]$
- **rej**(p, m) implies $p \notin [\![\varphi]\!]$

■

Soundness

Definition (Sound Monitoring)

Monitor m **soundly** monitors for the property φ, denoted as **smon**(m, φ), whenever for **every** system $p \in \mathrm{Sys}$ we have:

- **acc**(p, m) implies $p \in [\![\varphi]\!]$
- **rej**(p, m) implies $p \notin [\![\varphi]\!]$

■

Sound Monitors

- $\alpha.$✗ soundly monitors for $[\alpha]\mathbf{ff}$.
- $\alpha.$✓ $+ \beta.$✓ soundly monitors for $\langle\alpha\rangle\mathbf{tt} \vee \langle\beta\rangle\mathbf{tt}$.
- $\alpha.$✗ soundly monitors for $\mathbf{max}\, X.([\alpha]\mathbf{ff} \wedge [\beta]X)$.
- $\mathbf{rec}\, x.(\alpha.$✗ $+ \beta.x)$ soundly monitors for $\mathbf{max}\, X.([\alpha]\mathbf{ff} \wedge [\beta]X)$.

Quiz

Soundness Further Examples

Sound or Unsound?

① $\alpha.\checkmark$??? $[\alpha]\mathbf{ff}.$

Soundness Further Examples

Sound or Unsound?

① α.✓ **unsoundly** monitors for $[\alpha]$**ff**.

Soundness Further Examples

Sound or Unsound?

① $\alpha.\checkmark$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.

Witness $\alpha.\mathbf{nil}$: $\mathbf{acc}(\alpha.\mathbf{nil}, \alpha.\checkmark)$ but $\alpha.\mathbf{nil} \notin [\![[\alpha]\mathbf{ff}]\!]$.

Soundness Further Examples

Sound or Unsound?

① $\alpha.\checkmark$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.

Witness $\alpha.\mathbf{nil}$: $\mathbf{acc}(\alpha.\mathbf{nil}, \alpha.\checkmark)$ but $\alpha.\mathbf{nil} \notin [\![[\alpha]\mathbf{ff}]\!]$.

② $\alpha.\times$??? $[\alpha]\mathbf{ff} \wedge [\beta]\mathbf{ff}$.

Soundness Further Examples

Sound or Unsound?

① $\alpha.\checkmark$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.

Witness $\alpha.\mathbf{nil}$: $\mathbf{acc}(\alpha.\mathbf{nil}, \alpha.\checkmark)$ but $\alpha.\mathbf{nil} \notin [\![[\alpha]\mathbf{ff}]\!]$.

② $\alpha.\times$ **soundly** monitors for $[\alpha]\mathbf{ff} \wedge [\beta]\mathbf{ff}$.

Soundness Further Examples

Sound or Unsound?

① $\alpha.\checkmark$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.

Witness $\alpha.\mathbf{nil}$: $\mathbf{acc}(\alpha.\mathbf{nil}, \alpha.\checkmark)$ but $\alpha.\mathbf{nil} \notin [\![[\alpha]\mathbf{ff}]\!]$.

② $\alpha.\times$ **soundly** monitors for $[\alpha]\mathbf{ff} \wedge [\beta]\mathbf{ff}$.

③ $\alpha.\times + \beta.\times$??? $[\alpha]\mathbf{ff}$.

Soundness Further Examples

Sound or Unsound?

① $\alpha.\checkmark$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.

Witness $\alpha.\mathbf{nil}$: $\mathbf{acc}(\alpha.\mathbf{nil}, \alpha.\checkmark)$ but $\alpha.\mathbf{nil} \notin [\![[\alpha]\mathbf{ff}]\!]$.

② $\alpha.\times$ **soundly** monitors for $[\alpha]\mathbf{ff} \wedge [\beta]\mathbf{ff}$.

③ $\alpha.\times + \beta.\times$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.

Soundness Further Examples

Sound or Unsound?

① $\alpha.\checkmark$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.

Witness $\alpha.\mathbf{nil}$: $\mathbf{acc}(\alpha.\mathbf{nil}, \alpha.\checkmark)$ but $\alpha.\mathbf{nil} \notin [\![[\alpha]\mathbf{ff}]\!]$.

② $\alpha.\times$ **soundly** monitors for $[\alpha]\mathbf{ff} \wedge [\beta]\mathbf{ff}$.

③ $\alpha.\times + \beta.\times$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.
Consider the witness $\beta.\mathbf{nil}$

Soundness Further Examples

Sound or Unsound?

① $\alpha.\checkmark$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.

Witness $\alpha.\mathbf{nil}$: $\mathbf{acc}(\alpha.\mathbf{nil}, \alpha.\checkmark)$ but $\alpha.\mathbf{nil} \notin [\![[\alpha]\mathbf{ff}]\!]$.

② $\alpha.\times$ **soundly** monitors for $[\alpha]\mathbf{ff} \wedge [\beta]\mathbf{ff}$.

③ $\alpha.\times + \beta.\times$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.
Consider the witness $\beta.\mathbf{nil}$

④ $\alpha.\times + \beta.\checkmark$ **???** $[\alpha]\mathbf{ff}$.

Soundness Further Examples

Sound or Unsound?

① $\alpha.\checkmark$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.

Witness $\alpha.\mathbf{nil}$: $\mathbf{acc}(\alpha.\mathbf{nil}, \alpha.\checkmark)$ but $\alpha.\mathbf{nil} \notin [\![[\alpha]\mathbf{ff}]\!]$.

② $\alpha.\times$ **soundly** monitors for $[\alpha]\mathbf{ff} \wedge [\beta]\mathbf{ff}$.

③ $\alpha.\times + \beta.\times$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.
Consider the witness $\beta.\mathbf{nil}$

④ $\alpha.\times + \beta.\checkmark$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.

Soundness Further Examples

Sound or Unsound?

① $\alpha.\checkmark$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.

Witness $\alpha.\mathbf{nil}$: $\mathbf{acc}(\alpha.\mathbf{nil}, \alpha.\checkmark)$ but $\alpha.\mathbf{nil} \notin [\![[\alpha]\mathbf{ff}]\!]$.

② $\alpha.\times$ **soundly** monitors for $[\alpha]\mathbf{ff} \wedge [\beta]\mathbf{ff}$.

③ $\alpha.\times + \beta.\times$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.
Consider the witness $\beta.\mathbf{nil}$

④ $\alpha.\times + \beta.\checkmark$ **unsoundly** monitors for $[\alpha]\mathbf{ff}$.
Consider the witness $\alpha.\mathbf{nil} + \beta.\mathbf{nil}$

Sound Monitors and Inconsistent Verdicts

Definition

- $\textbf{sound}(m) \stackrel{\text{def}}{=} \exists \varphi \cdot \text{smon}(m, \varphi)$
- $\textbf{incons}(m) \stackrel{\text{def}}{=} \exists p \cdot \textbf{acc}(p, m)$ and $\textbf{rej}(p, m)$.

Theorem

sound(m) *implies* $\neg$***incons***(m)

Sound Monitors and Inconsistent Verdicts

Definition

- $\textbf{sound}(m) \stackrel{\text{def}}{=} \exists \varphi \cdot \text{smon}(m, \varphi)$
- $\textbf{incons}(m) \stackrel{\text{def}}{=} \exists p \cdot \textbf{acc}(p, m)$ and $\textbf{rej}(p, m)$.

Theorem

sound(m) *implies* $\neg$***incons***(m)

Proof.

sound(m)	$\Rightarrow \exists \varphi \cdot \text{smon}(m, \varphi)$
Assume $p \in [\![\varphi]\!]$	
Semantics of $[\![\varphi]\!]$	$\Rightarrow \neg(p \notin [\![\varphi]\!])$
smon$(m, \varphi)$and Contrap.	$\Rightarrow \neg(\textbf{rej}(p, m))$

Sound Monitors and Inconsistent Verdicts

Definition

- $\textbf{sound}(m) \stackrel{\text{def}}{=} \exists \varphi \cdot \text{smon}(m, \varphi)$
- $\textbf{incons}(m) \stackrel{\text{def}}{=} \exists p \cdot \textbf{acc}(p, m)$ and $\textbf{rej}(p, m)$.

Theorem

sound(m) *implies* $\neg$***incons***(m)

Soundness is Weak

- Monitor α.✗ soundly monitors for $[\alpha]\textbf{ff} \wedge [\beta]\textbf{ff}$.
- Monitor **?** soundly monitors for every $\varphi \in \mu\text{HML}$ but it is *not* very useful!

Verdicts Vs Satisfactions and Violations

Completeness Predicates

Definition (Satifaction and Violation Complete)

$$\text{scmon}(m, \varphi) \stackrel{\text{def}}{=} \forall p.\, p \in [\![\varphi]\!] \quad \text{implies} \quad \textbf{acc}(p, m)$$

$$\text{vcmon}(m, \varphi) \stackrel{\text{def}}{=} \forall p.\, p \notin [\![\varphi]\!] \quad \text{implies} \quad \textbf{rej}(p, m)$$

Completeness Predicates

Definition (Satifaction and Violation Complete)

$$\text{scmon}(m, \varphi) \stackrel{\text{def}}{=} \forall p.\, p \in [\![\varphi]\!] \quad \text{implies} \quad \textbf{acc}(p, m)$$

$$\text{vcmon}(m, \varphi) \stackrel{\text{def}}{=} \forall p.\, p \notin [\![\varphi]\!] \quad \text{implies} \quad \textbf{rej}(p, m)$$

No sound and violation-complete monitor for $\langle\alpha\rangle$tt

Completeness Predicates

Definition (Satifaction and Violation Complete)

$$\text{scmon}(m, \varphi) \stackrel{\text{def}}{=} \forall p.\, p \in [\![\varphi]\!] \quad \text{implies} \quad \mathbf{acc}(p, m)$$

$$\text{vcmon}(m, \varphi) \stackrel{\text{def}}{=} \forall p.\, p \notin [\![\varphi]\!] \quad \text{implies} \quad \mathbf{rej}(p, m)$$

No sound and violation-complete monitor for $\langle\alpha\rangle$tt

Proof Outline: Assume one such monitor m exists.
Since $\mathbf{nil} \notin [\![\langle\alpha\rangle\mathbf{tt}]\!]$ then $\mathbf{rej}(\mathbf{nil}, m)$.

Completeness Predicates

Definition (Satifaction and Violation Complete)

$$\text{scmon}(m, \varphi) \stackrel{\text{def}}{=} \forall p.\, p \in [\![\varphi]\!] \quad \text{implies} \quad \mathbf{acc}(p, m)$$

$$\text{vcmon}(m, \varphi) \stackrel{\text{def}}{=} \forall p.\, p \notin [\![\varphi]\!] \quad \text{implies} \quad \mathbf{rej}(p, m)$$

No sound and violation-complete monitor for $\langle\alpha\rangle$tt

Proof Outline: Assume one such monitor m exists.
Since $\mathbf{nil} \notin [\![\langle\alpha\rangle\mathbf{tt}]\!]$ then $\mathbf{rej}(\mathbf{nil}, m)$.
Monitor m must reach ✗ independent of observations.

Completeness Predicates

Definition (Satifaction and Violation Complete)

$$\text{scmon}(m, \varphi) \stackrel{\text{def}}{=} \forall p.\, p \in [\![\varphi]\!] \quad \text{implies} \quad \mathbf{acc}(p, m)$$

$$\text{vcmon}(m, \varphi) \stackrel{\text{def}}{=} \forall p.\, p \notin [\![\varphi]\!] \quad \text{implies} \quad \mathbf{rej}(p, m)$$

No sound and violation-complete monitor for $\langle\alpha\rangle$tt

Proof Outline: Assume one such monitor m exists.
Since $\mathbf{nil} \notin [\![\langle\alpha\rangle\mathbf{tt}]\!]$ then $\mathbf{rej}(\mathbf{nil}, m)$.
Monitor m must reach ✗ independent of observations.
Then $\mathbf{rej}(\alpha.\mathbf{nil}, m)$ as well.
This is **unsound** because $\alpha.\mathbf{nil} \in [\![\langle\alpha\rangle\mathbf{tt}]\!]$.

Completeness Predicates

Definition (Satifaction and Violation Complete)

$$\text{scmon}(m, \varphi) \stackrel{\text{def}}{=} \forall p.\, p \in [\![\varphi]\!] \quad \text{implies} \quad \mathbf{acc}(p, m)$$

$$\text{vcmon}(m, \varphi) \stackrel{\text{def}}{=} \forall p.\, p \notin [\![\varphi]\!] \quad \text{implies} \quad \mathbf{rej}(p, m)$$

No sound and violation-complete monitor for $\langle\alpha\rangle$tt

Proof Outline: Assume one such monitor m exists.
Since $\mathbf{nil} \notin [\![\langle\alpha\rangle\mathbf{tt}]\!]$ then $\mathbf{rej}(\mathbf{nil}, m)$.
Monitor m must reach ✗ independent of observations.
Then $\mathbf{rej}(\alpha.\mathbf{nil}, m)$ as well.
This is **unsound** because $\alpha.\mathbf{nil} \in [\![\langle\alpha\rangle\mathbf{tt}]\!]$.

!!No sound **and** satisfaction-complete monitor for $[\alpha]\mathbf{ff}$ either!!

Runtime Verification Mantra

Half a loaf is better than no bread

(Something is always *better than nothing)*

Monitor-Formula Correspondence (Correctness)

Definition (Partial-Completeness)

A monitor m monitors for formula φ in a **partially-complete** manner, denoted as **cmon**(m, φ), when either scmon(m, φ) **or** vcmon(m, φ) holds.

Definition (Correct Monitors)

A monitor m is said to **monitor for** a formula φ (correctly), denoted as mon(m, φ), if it can do it **soundly**, and in a **partially-complete** manner:

$$\textbf{mon}(m, \varphi) \stackrel{\text{def}}{=} \text{smon}(m, \varphi) \text{ and } \text{cmon}(m, \varphi)$$

Quiz

Monitors For Examples

Does it monitor for it?

1. α.✗ **monitors for** $[\alpha]$**ff**.
2. α.✓ **monitors for** $\langle\alpha\rangle$**tt**.

Monitors For Examples

Does it monitor for it?

① α.✗ **monitors for** $[\alpha]\mathbf{ff}$.

② α.✓ **monitors for** $\langle\alpha\rangle\mathbf{tt}$.

③ α.✗ ??? $[\alpha]\mathbf{ff} \wedge [\beta]\mathbf{ff}$.

Monitors For Examples

Does it monitor for it?

1. $\alpha.\text{✗}$ **monitors for** $[\alpha]\mathbf{ff}$.
2. $\alpha.\text{✓}$ **monitors for** $\langle\alpha\rangle\mathbf{tt}$.

3. $\alpha.\text{✗}$ **does not** monitor for $[\alpha]\mathbf{ff} \wedge [\beta]\mathbf{ff}$.
4. $\alpha.\text{✗} + \beta.\text{✗}$ **monitor for** $[\alpha]\mathbf{ff} \wedge [\beta]\mathbf{ff}$.

Monitors For Examples

Does it monitor for it?

① $\alpha.\text{✗}$ **monitors for** $[\alpha]\mathbf{ff}$.

② $\alpha.\text{✓}$ **monitors for** $\langle\alpha\rangle\mathbf{tt}$.

③ $\alpha.\text{✗}$ **does not** monitor for $[\alpha]\mathbf{ff} \wedge [\beta]\mathbf{ff}$.

④ $\alpha.\text{✗} + \beta.\text{✗}$ **monitor for** $[\alpha]\mathbf{ff} \wedge [\beta]\mathbf{ff}$.

⑤ $\alpha.\text{✗}$??? $[\alpha][\beta]\mathbf{ff}$.

Monitors For Examples

Does it monitor for it?

① $\alpha.\checkmark$ **monitors for** $[\alpha]\mathbf{ff}$.

② $\alpha.\checkmark$ **monitors for** $\langle\alpha\rangle\mathbf{tt}$.

③ $\alpha.\checkmark$ **does not** monitor for $[\alpha]\mathbf{ff} \wedge [\beta]\mathbf{ff}$.

④ $\alpha.\checkmark + \beta.\checkmark$ **monitor for** $[\alpha]\mathbf{ff} \wedge [\beta]\mathbf{ff}$.

⑤ $\alpha.\checkmark$ **does not** monitor for $[\alpha][\beta]\mathbf{ff}$.

⑥ $\alpha.\beta.\checkmark$ **monitor for** $[\alpha][\beta]\mathbf{ff}$.

Monitorability

Definition (Monitorability)

- A formula $\varphi \in \mu\text{HML}$ is **monitorable** iff there **exists** a monitor m such that **mon**(m, φ).

Monitorability

Definition (Monitorability)

- A formula $\varphi \in \mu\text{HML}$ is **monitorable** iff there **exists** a monitor m such that **mon**(m, φ).
- A **set of formulae** $\mathcal{L} \subseteq \mu\text{HML}$ is monitorable iff **every** formula in the set, $\varphi \in \mathcal{L}$, is monitorable.

Monitorability

Definition (Monitorability)

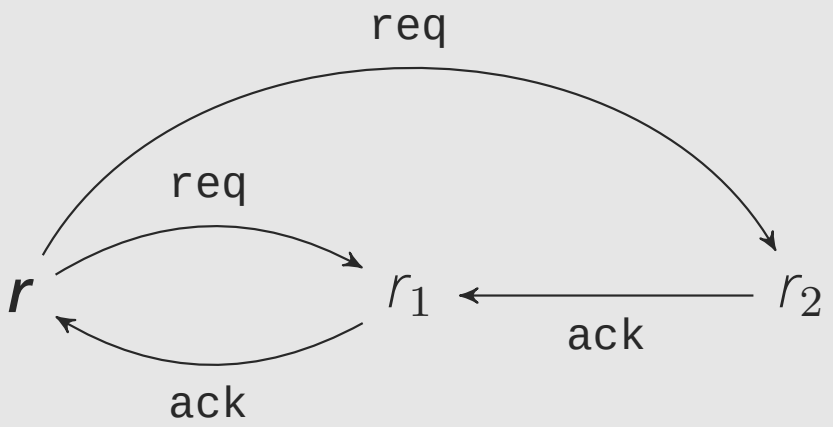
- A formula $\varphi \in \mu\text{HML}$ is **monitorable** iff there **exists** a monitor m such that **mon**(m, φ).
- A **set of formulae** $\mathcal{L} \subseteq \mu\text{HML}$ is monitorable iff **every** formula in the set, $\varphi \in \mathcal{L}$, is monitorable.

Showing that a (sub)logic $\mathcal{L}$ is monitorable requires two universal quantifications:

1. Over all $\varphi \in \mathcal{L}$.
2. Over all $p \in [\![\varphi]\!]$.

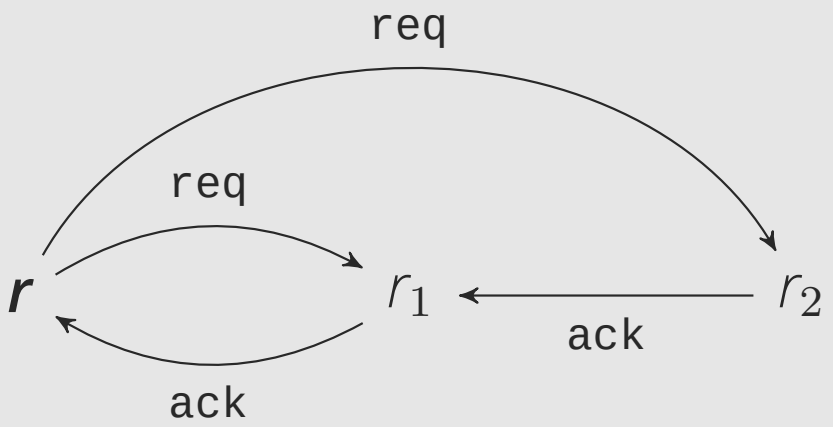
Quiz

Monitorable and Non-monitorable Formulas



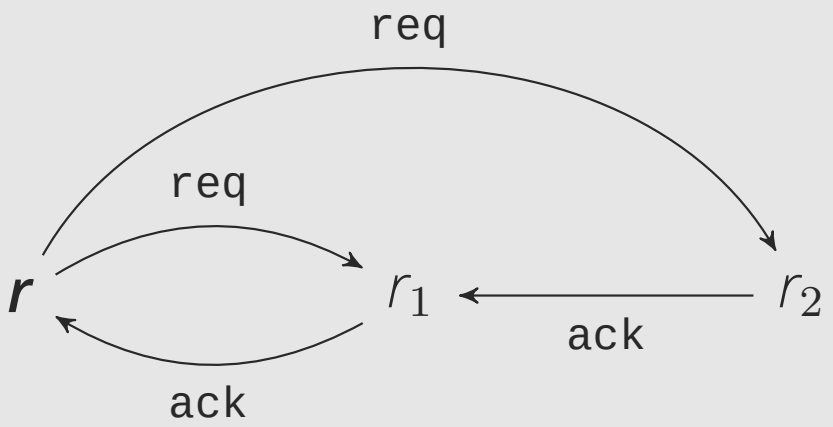
$\langle \mathtt{req} \rangle \mathbf{tt}$

Monitorable and Non-monitorable Formulas



$$r \in [\![\langle \mathrm{req} \rangle \mathbf{tt}]\!]$$

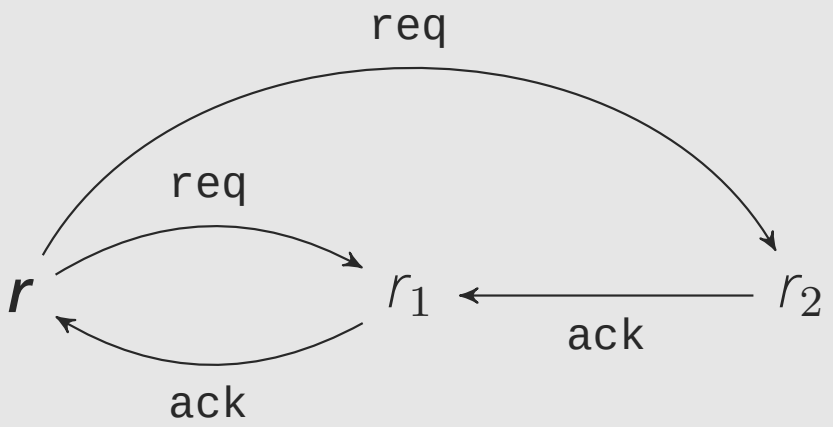
Monitorable and Non-monitorable Formulas



$r \in [\![\langle \text{req} \rangle \mathbf{tt}]\!]$

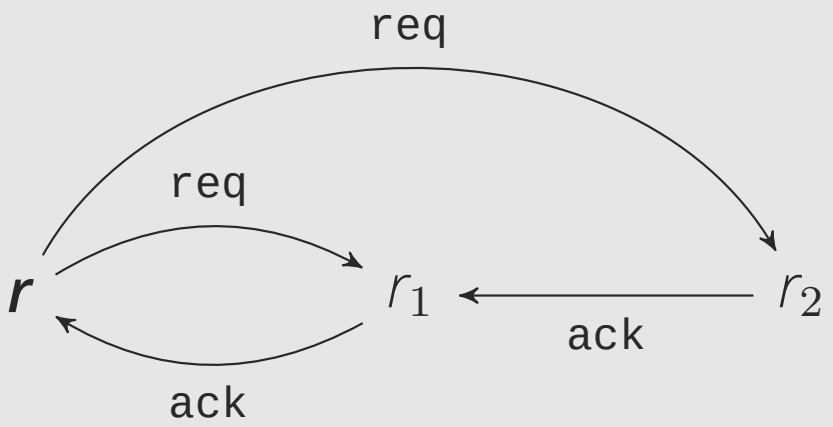
$m = \text{req}.$✓

Monitorable and Non-monitorable Formulas



$r_1 \notin [\![\langle \text{req} \rangle \mathbf{tt}]\!]$

Monitorable and Non-monitorable Formulas



$r_1 \notin [\![\langle \mathsf{req} \rangle \mathbf{tt}]\!]$

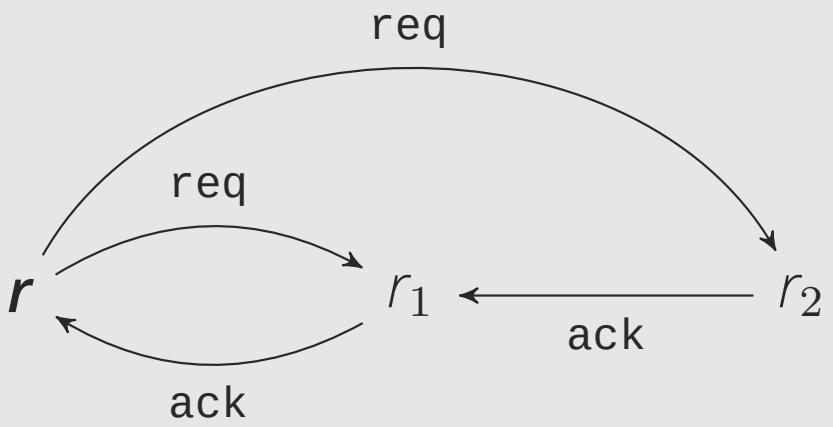
$m = \mathbf{??}$

Runtime Verification Mantra

Half a loaf is better than no bread

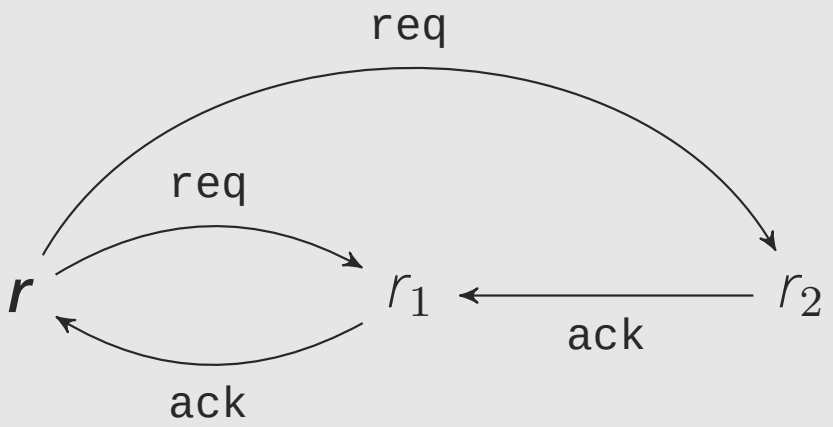
(Something is always *better than nothing)*

Monitorable and Non-monitorable Formulas



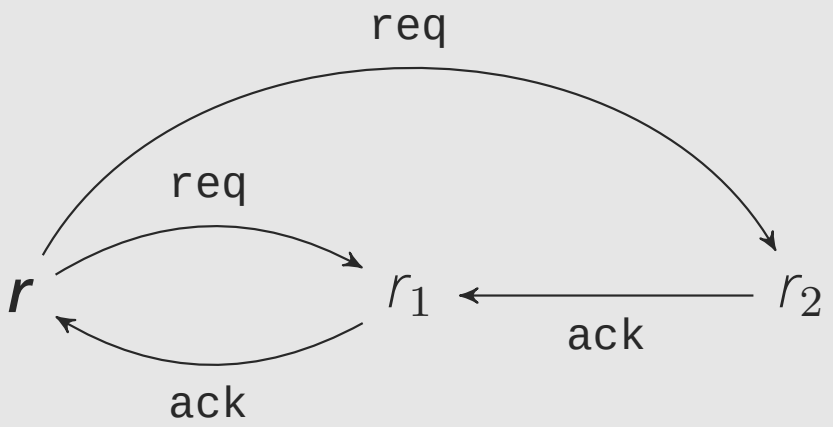
[ack]**ff**

Monitorable and Non-monitorable Formulas



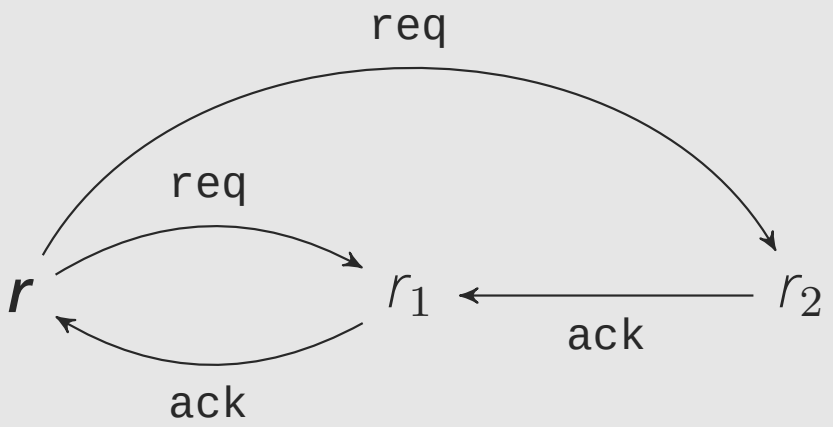
$r \in [\![[\text{ack}]\mathbf{ff}]\!]$

Monitorable and Non-monitorable Formulas



$r \in [\![[\text{ack}]\mathbf{ff}]\!]$ $\qquad$ $m = \mathbf{??}$

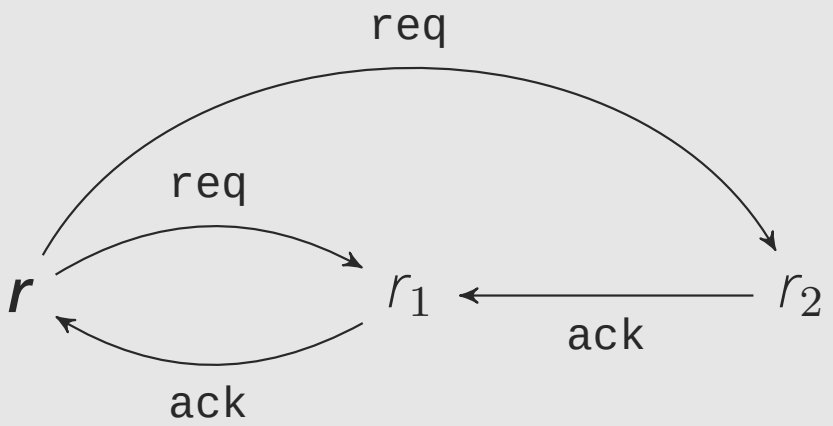
Monitorable and Non-monitorable Formulas



$r_1 \notin [\![[\text{ack}]\mathbf{ff}]\!]$

Monitorability

Monitorable and Non-monitorable Formulas



$r_1 \notin [\![[\text{ack}]\mathbf{ff}]\!]$

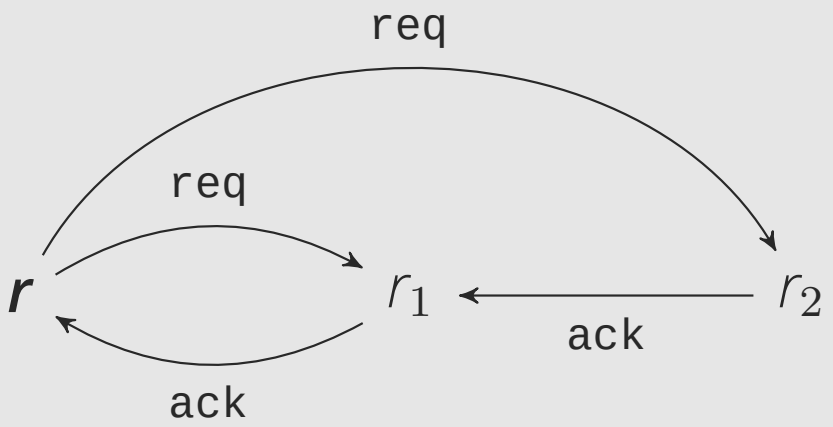
$m = \text{ack}.$ ✗

Runtime Verification Mantra

Half a loaf is better than no bread

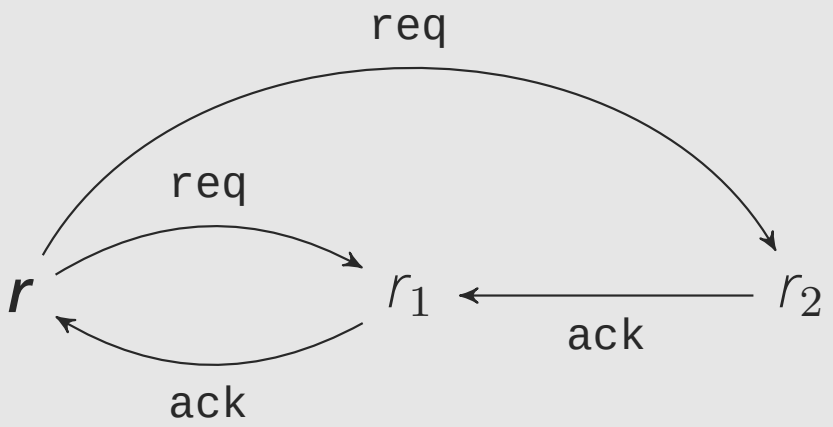
(Something is always *better than nothing)*

Monitorable and Non-monitorable Formulas



$$[\mathtt{req}]\langle \mathtt{ack} \rangle \mathbf{tt}$$

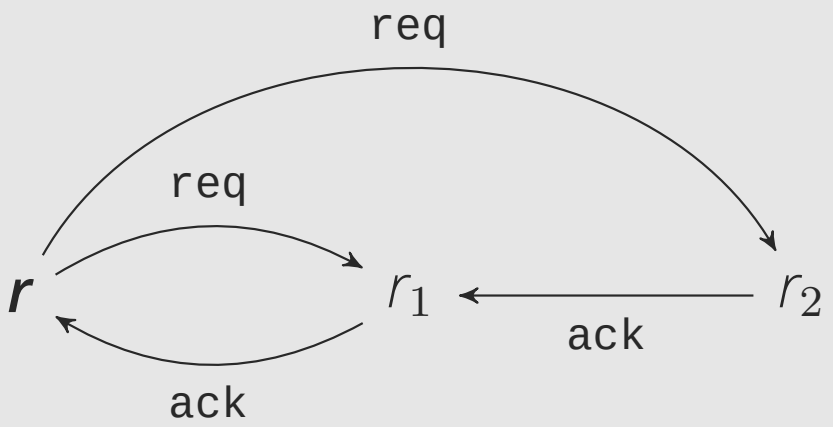
Monitorable and Non-monitorable Formulas



$r \in [\![[\text{req}]\langle\text{ack}\rangle \mathbf{tt}]\!]$

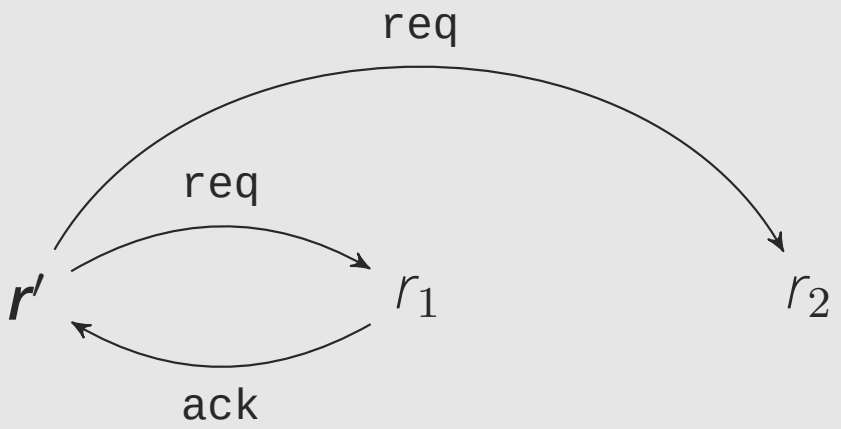
Monitorability

Monitorable and Non-monitorable Formulas



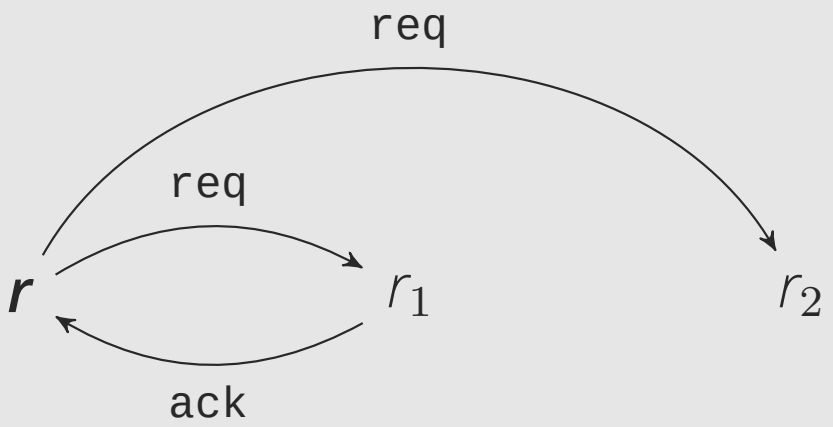
$r \in [\![[\text{req}]\langle\text{ack}\rangle \mathbf{tt}]\!]$ $\qquad$ $m = \mathbf{??}$

Monitorable and Non-monitorable Formulas



$r' \notin [\![[\mathrm{req}] \langle \mathrm{ack} \rangle \mathbf{tt}]\!]$

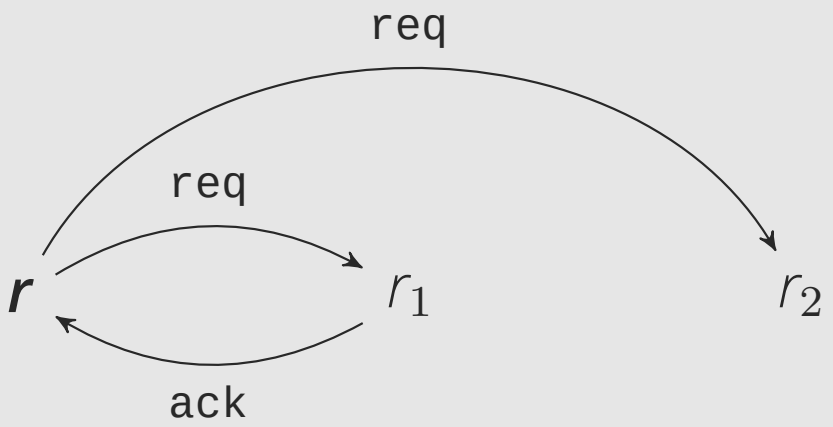
Monitorable and Non-monitorable Formulas



$r' \notin [\![[\text{req}]\langle\text{ack}\rangle \mathbf{tt}]\!]$

$m = \mathbf{??}$

Monitorable and Non-monitorable Formulas



$[\![[\text{req}]\langle\text{ack}\rangle\mathbf{tt}]\!]$

Runtime Verification Mantra

Half a loaf is better than no bread

(Something is always *better than nothing*
...but there not much we can do)

Monitorability

Not all formulas are monitorable

Consider

$$\varphi = [\alpha]\mathbf{ff} \vee [\beta]\mathbf{ff}$$

Monitorability

Not all formulas are monitorable

Consider

$$\varphi = [\alpha]\mathbf{ff} \vee [\beta]\mathbf{ff}$$

and assume $\exists m \cdot \mathrm{mon}(m, \varphi)$. Then m must be **sound** and

Monitorability

Not all formulas are monitorable

Consider

$$\varphi = [\alpha]\mathbf{ff} \vee [\beta]\mathbf{ff}$$

and assume $\exists m \cdot \mathrm{mon}(m, \varphi)$. Then m must be **sound** and

Satisfaction Complete: Since $\beta.\mathbf{0} \in [\![[\alpha]\mathbf{ff} \vee [\beta]\mathbf{ff}]\!]$ we should have $\mathbf{acc}(\beta.\mathbf{0}, m)$. But then we must also have $\mathbf{acc}(\alpha.\mathbf{0} + \beta.\mathbf{0}, m)$, which is unsound.

Monitorability

Not all formulas are monitorable

Consider

$$\varphi = [\alpha]\mathbf{ff} \vee [\beta]\mathbf{ff}$$

and assume $\exists m \cdot \mathrm{mon}(m, \varphi)$. Then m must be **sound** and

Satisfaction Complete: Since $\beta.\mathbf{0} \in [\![[\alpha]\mathbf{ff} \vee [\beta]\mathbf{ff}]\!]$ we should have $\mathbf{acc}(\beta.\mathbf{0}, m)$. But then we must also have $\mathbf{acc}(\alpha.\mathbf{0} + \beta.\mathbf{0}, m)$, which is unsound.

Violation Complete: Since $\alpha.\mathbf{0} + \beta.\mathbf{0} \notin [\![[\alpha]\mathbf{ff} \vee [\beta]\mathbf{ff}]\!]$ we should have $\mathbf{rej}(\alpha.\mathbf{0} + \beta.\mathbf{0}, m)$. This means that it must either reject **nil**, $\beta.\mathbf{0}$ or $\alpha.\mathbf{0}$, which is unsound.

A Maximally Monitorable Fragment

Definition (Monitorable Logic)

Let $\mathrm{mHML} = \mathrm{cHML} \cup \mathrm{sHML}$, where

$$\pi, \varpi \in \mathrm{cHML} ::= \mathbf{tt} \mid \mathbf{ff} \mid \pi \vee \varpi \mid \langle\alpha\rangle\pi \mid \mathbf{min}\, X.\pi \mid X$$

$$\theta, \vartheta \in \mathrm{sHML} ::= \mathbf{tt} \mid \mathbf{ff} \mid \theta \wedge \vartheta \mid [\alpha]\theta \mid \mathbf{max}\, X.\theta \mid X$$

A Maximally Monitorable Fragment

Definition (Monitorable Logic)

Let $\mathrm{mHML} = \mathrm{cHML} \cup \mathrm{sHML}$, where

$$\pi, \varpi \in \mathrm{cHML} ::= \mathbf{tt} \mid \mathbf{ff} \mid \pi \vee \varpi \mid \langle\alpha\rangle\pi \mid \mathbf{min}\, X.\pi \mid X$$
$$\theta, \vartheta \in \mathrm{sHML} ::= \mathbf{tt} \mid \mathbf{ff} \mid \theta \wedge \vartheta \mid [\alpha]\theta \mid \mathbf{max}\, X.\theta \mid X$$

Main Results [FAI17]

- mHML is monitorable.
- mHML is maximally expressive.

How is this useful?

1. Monitor Correctness [FAI15, FAI17, FAA+17]
2. Property specification [FAI15, FAI17, FAA+17]

How is this useful?

1. Monitor Correctness [FAI15, FAI17, FAA+17]
2. Property specification [FAI15, FAI17, FAA+17]
3. Understanding monitorability for other logics.

How is this useful?

1. Monitor Correctness [FAI15, FAI17, FAA+17]
2. Property specification [FAI15, FAI17, FAA+17]
3. Understanding monitorability for other logics.
4. Tool construction [AF16, FS15].

How is this useful?

1. Monitor Correctness [FAI15, FAI17, FAA+17]
2. Property specification [FAI15, FAI17, FAA+17]
3. Understanding monitorability for other logics.
4. Tool construction [AF16, FS15].
5. Program verification management [DMF15].

$$\langle\alpha\rangle(\langle\beta\rangle\mathbf{tt} \vee [\gamma]\mathbf{ff}) \xrightarrow{\text{refactored as}} (\langle\alpha\rangle\langle\beta\rangle\mathbf{tt}) \vee (\langle\alpha\rangle[\gamma]\mathbf{ff})$$

How is this useful?

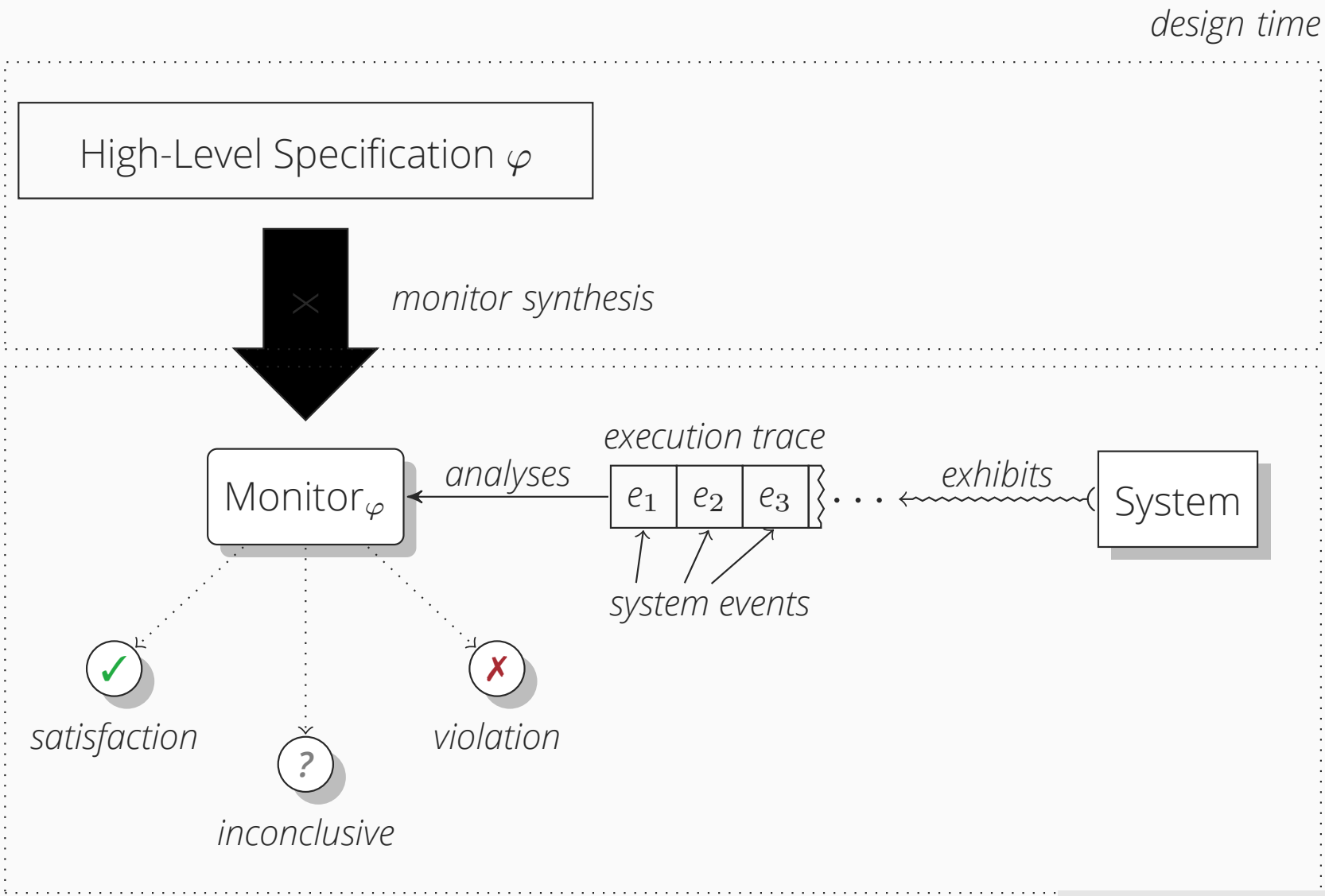
1. Monitor Correctness [FAI15, FAI17, FAA$^+$17]
2. Property specification [FAI15, FAI17, FAA$^+$17]
3. Understanding monitorability for other logics.
4. Tool construction [AF16, FS15].
5. Program verification management [DMF15].

$$\langle\alpha\rangle(\langle\beta\rangle\mathbf{tt} \vee [\gamma]\mathbf{ff}) \xrightarrow{\text{refactored as}} (\langle\alpha\rangle\langle\beta\rangle\mathbf{tt}) \vee (\langle\alpha\rangle[\gamma]\mathbf{ff})$$

6. Extending monitorability [AF17, AAFI17, AAFI18]

To Monitorabilty...and Beyond!

Know Your Limits



Know Your Limits

Definition (Monitorable Logic)

Let $\mathrm{mHML} = \mathrm{cHML} \cup \mathrm{sHML}$, where

$$\pi, \varpi \in \mathrm{cHML} ::= \mathbf{tt} \mid \mathbf{ff} \mid \pi \vee \varpi \mid \langle\alpha\rangle\pi \mid \mathbf{min}\, X.\pi \mid X$$
$$\theta, \vartheta \in \mathrm{sHML} ::= \mathbf{tt} \mid \mathbf{ff} \mid \theta \wedge \vartheta \mid [\alpha]\theta \mid \mathbf{max}\, X.\theta \mid X$$

Know Your Limits

Definition (Monitorable Logic)

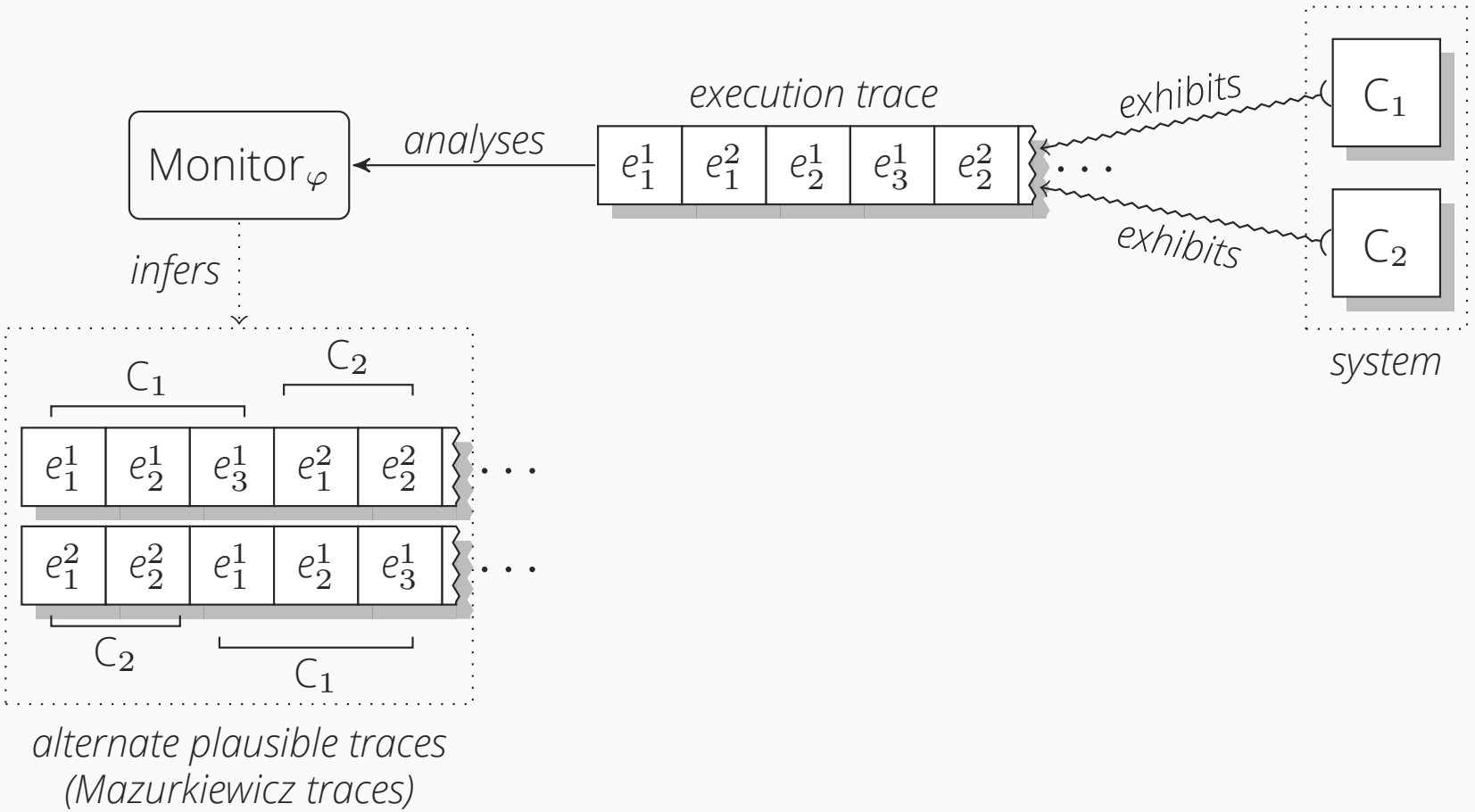
Let $\mathrm{mHML} = \mathrm{cHML} \cup \mathrm{sHML}$, where

$$\pi, \varpi \in \mathrm{cHML} ::= \mathbf{tt} \mid \mathbf{ff} \mid \pi \vee \varpi \mid \langle\alpha\rangle\pi \mid \mathbf{min}\, X.\pi \mid X$$
$$\theta, \vartheta \in \mathrm{sHML} ::= \mathbf{tt} \mid \mathbf{ff} \mid \theta \wedge \vartheta \mid [\alpha]\theta \mid \mathbf{max}\, X.\theta \mid X$$

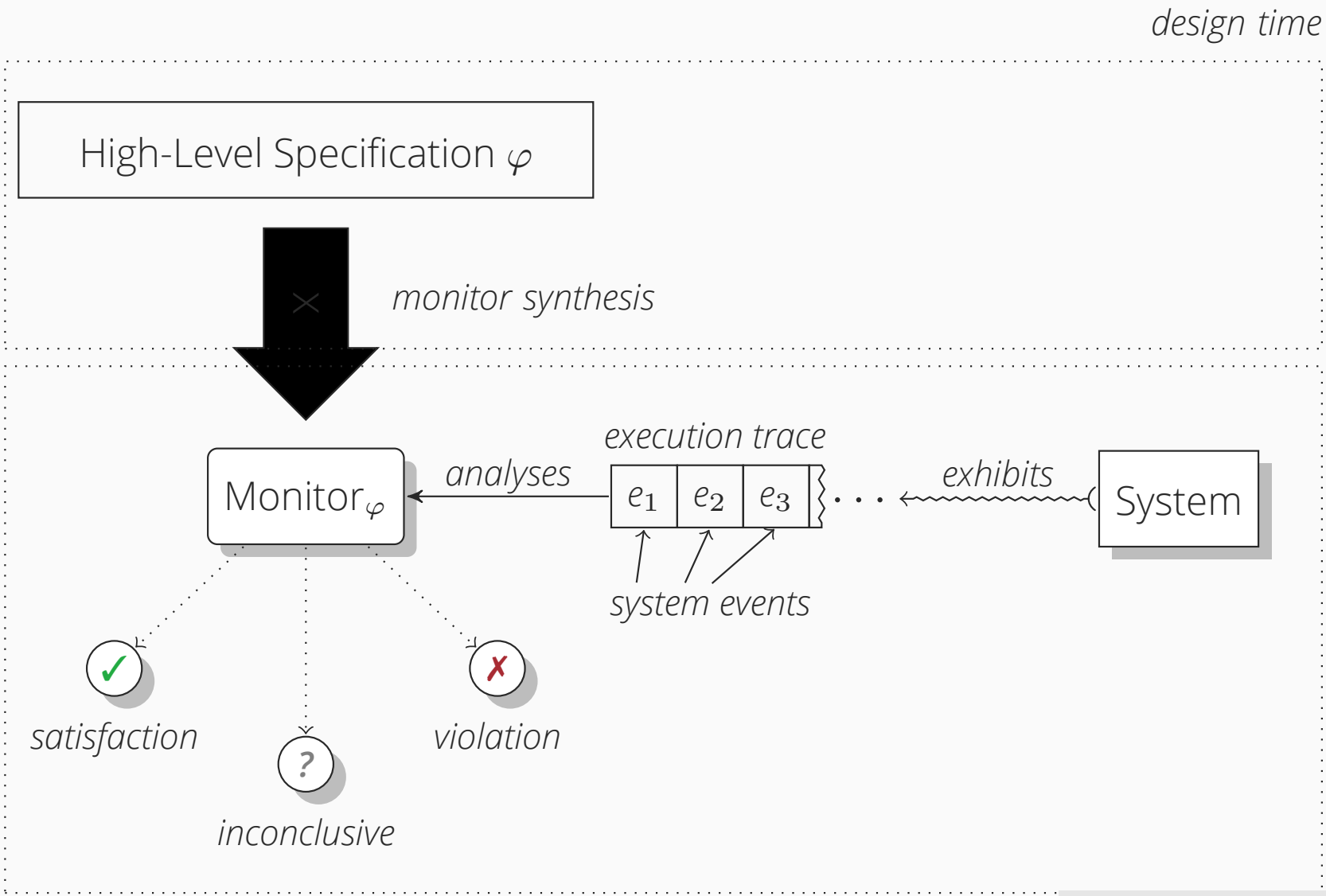
"So...is this it?"

- ponderings of a forty-something

No Limits?


Monitor$_\varphi$
analyses
execution trace
exhibits
exhibits
C_1
C_2
system
infers
C_1
C_2
C_2
C_1
alternate plausible traces
(Mazurkiewicz traces)

Revisiting the Classical Monitoring Setup



What Have these Traces Ever Done for Us?

- They report the events that **have happened**.

What Have these Traces Ever Done for Us?

- They report the events that **have happened**.
- They could also report the events that **could not have happened** (at specific points in an execution).

What Have these Traces Ever Done for Us?

- They report the events that **have happened**.
- They could also report the events that **could not have happened** (at specific points in an execution).
- They could include information from logs **from previously monitored runs**.

What Have these Traces Ever Done for Us?

- They report the events that **have happened**.
- They could also report the events that **could not have happened** (at specific points in an execution).
- They could include information from logs **from previously monitored runs**.
- They could also include **descriptions of (depth bounded) trees of computation** at specific points in an execution.

Extended Monitoring Setup

Definition (Extended Monitors)

$$m, n \in \text{Mon} \ ::= \ \ldots \quad \mid !\alpha.m$$

$$\text{Mon}\ \frac{p \xrightarrow{\alpha} p' \qquad m \xrightarrow{\alpha} m'}{m \triangleleft p \xrightarrow{\alpha} m' \triangleleft p'}$$

$$\text{Ter}\ \frac{p \xrightarrow{\alpha} p' \qquad m \not\xrightarrow{\alpha}}{m \triangleleft p \xrightarrow{\alpha} ? \triangleleft p'}$$

$$\text{Neg}\ \frac{p \not\xrightarrow{\alpha} p' \qquad m \xrightarrow{!\alpha} m'}{m \triangleleft p \xrightarrow{\tau} m' \triangleleft p'}$$

Extending the Monitorable Fragment

Recall the safety (monitorable) fragment sHML:

$$\theta, \vartheta \in \text{sHML} ::= \mathbf{tt} \quad | \; \mathbf{ff} \quad | \; \theta \wedge \vartheta \quad | \; [\alpha]\theta \quad | \; \mathbf{max} X.\theta \quad | \; X$$

Extending the Monitorable Fragment

Recall the safety (monitorable) fragment sHML:

$$\theta, \vartheta \in \text{sHML} ::= \mathbf{tt} \quad | \ \mathbf{ff} \quad | \ \theta \wedge \vartheta \quad | \ [\alpha]\theta \quad | \ \mathbf{max} X.\theta \quad | \ X$$

In the extended framework, this (maximal) monitorable fragment extends to:

$$\theta, \vartheta \in \text{sHML}^{+} ::= \mathbf{tt} \quad | \ \mathbf{ff} \quad | \ \theta \wedge \vartheta \quad | \ [\alpha]\theta \quad | \ \langle\alpha\rangle\mathbf{tt} \vee \theta \quad | \ \mathbf{max} X.\theta \quad | \ X$$

Extended Monitoring Setup (2)

Definition (Extended Monitors)

$$\eta, \chi \in \mathit{HML} ::= \mathbf{tt} \mid \mathbf{ff} \mid \eta \wedge \chi \mid \eta \vee \chi \mid [\alpha]\eta \mid \langle\alpha\rangle\eta$$
$$m, n \in \mathrm{Mon} ::= \ldots \mid \eta.m$$

$$\mathrm{Mon}\ \frac{p \xrightarrow{\alpha} p' \quad m \xrightarrow{\alpha} m'}{m \triangleleft p \xrightarrow{\alpha} m' \triangleleft p'}$$

$$\mathrm{Ter}\ \frac{p \xrightarrow{\alpha} p' \quad m \not\xrightarrow{\alpha}}{m \triangleleft p \xrightarrow{\alpha} \mathbf{?} \triangleleft p'}$$

$$\mathrm{Frm}\ \frac{p \in [\![\eta]\!] \quad m \xrightarrow{\eta} m'}{m \triangleleft p \xrightarrow{\tau} m' \triangleleft p'}$$

Extending the Monitorable Fragment (2)

Recall the safety (monitorable) fragment sHML:

$$\theta, \vartheta \in \text{sHML} ::= \mathbf{tt} \quad | \ \mathbf{ff} \quad | \ \theta \wedge \vartheta \quad | \ [\alpha]\theta \quad | \ \mathbf{max} X.\theta \quad | \ X$$

Extending the Monitorable Fragment (2)

Recall the safety (monitorable) fragment sHML:

$$\theta, \vartheta \in \text{sHML} ::= \mathbf{tt} \quad | \ \mathbf{ff} \quad | \ \theta \wedge \vartheta \quad | \ [\alpha]\theta \quad | \ \mathbf{max} X.\theta \quad | \ X$$

In the extended framework, this (maximal) monitorable fragment extends to:

$$\begin{aligned} \theta, \vartheta \in \text{sHML}^{++} ::= \ & \mathbf{tt} \quad | \ \mathbf{ff} \quad | \ \theta \wedge \vartheta \quad | \ [\alpha]\theta \\ & | \ \eta \vee \theta \quad | \ \mathbf{max} X.\theta \quad | \ X \end{aligned}$$

Additional Work

- Tool Construction:
  - Automated monitor synthesis [AF16, CF16].
  - Language-Specific instrumentation tools [CFAI17].
- Complexity Results for Monitor Determinisation [AAF+17].
- Automated Formula Refactoring to isolate monitorable parts [DMF15].
- (Correctness) Monitor Preorders for refinement [Fra16].
- (Correctness) Monitor Properties that extend to reactive software in general [Fra17].

Contributors

Luca Aceto, Antonis Achilleos, Duncan Paul Attard, Ian Cassar, Dario Della Monica, Anna Ingolfsdottir, myself (and more researchers will be joining us soon).

What now?

"Verifiers of the world, unite!
You have nothing to lose,
but your own preconceived
verification shackles!"

Further Reading

- The first part of the talk is based on the high-level tutorial paper [FAA+17] available **here**.
- For an in-depth treatment of the topics, consult [FAI17] **here**.
- For the topics discussed in the last section, consult [AAFI18] **here**.
- For a full overview of the work conducted in the TheoFoMon Project, consult **here**.

References I

Luca Aceto, Antonis Achilleos, Adrian Francalanza, Anna Ingólfsdóttir, and Sævar Örn Kjartansson.
On the Complexity of Determinizing Monitors.
In *CIAA*, volume 10329 of *LNCS*, pages 1–13, 2017.

Luca Aceto, Antonis Achilleos, Adrian Francalanza, and Anna Ingólfsdóttir.
Monitoring for silent actions.
In *FSTTCS*, volume 93 of *LIPIcs*, pages 7:1–7:14, 2017.

Luca Aceto, Antonis Achilleos, Adrian Francalanza, and Anna Ingólfsdóttir.
A Framework for Parametrized Monitorability.
In *FoSSaCS*, LNCS, 2018.
(to appear).

Duncan Paul Attard and Adrian Francalanza.
A Monitoring Tool for a Branching-Time Logic.
In *RV*, volume 10012 of *LNCS*, pages 473–481, 2016.

Duncan Paul Attard and Adrian Francalanza.
Trace Partitioning and Local Monitoring for Asynchronous Components.
In *SEFM*, LNCS, 2017.
(to appear).

Ian Cassar and Adrian Francalanza.
On Implementing a Monitor-Oriented Programming Framework for Actor Systems.
In *IFM*, volume 9681 of *LNCS*, pages 176–192, 2016.

References II

Ian Cassar, Adrian Francalanza, Luca Aceto, and Anna Ingólfsdóttir.

eAOP - An Aspect Oriented Programming Framework for Erlang.
In *Erlang Workshop*, 2017.
(to appear).

Dario Della Monica and Adrian Francalanza.

Towards a Hybrid Approach to Software Verification.
In *NWPT*, number SCS16001 in RUTR, pages 51–54, 2015.

Adrian Francalanza, Luca Aceto, Antonis Achilleos, Duncan Paul Attard, Ian Cassar, Dario Della Monica, and Anna Ingólfsdóttir.

A Foundation for Runtime Monitoring.
In *Runtime Verification: 17th International Conference, RV 2017, Seattle, WA, USA, September 2017*, volume 10548 of *LNCS*, pages 8–29. Springer, 2017.

Adrian Francalanza, Luca Aceto, and Anna Ingólfsdóttir.

On Verifying Hennessy-Milner Logic with Recursion at Runtime.
In *RV*, volume 9333 of *LNCS*, pages 71–86, 2015.

Adrian Francalanza, Luca Aceto, and Anna Ingolfsdottir.

Monitorability for the Hennessy–Milner Logic with Recursion.
Formal Methods in System Design, pages 1–30, 2017.

Adrian Francalanza.

A Theory of Monitors - (Extended Abstract).
In *FoSSaCS*, volume 9634 of *LNCS*, pages 145–161, 2016.

References III

Adrian Francalanza.

Consistently-Detecting Monitors.
In Roland Meyer and Uwe Nestmann, editors, *CONCUR*, volume 85 of *LIPIcs*, pages 8:1–8:19, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Adrian Francalanza and Aldrin Seychell.

Synthesising Correct Concurrent Runtime Monitors.
Formal Methods in System Design, 46(3):226–261, 2015.