# Topic: GR(1) Synthesis

Lecture at the
$1^{st}$ Summer School on Formal Methods for Cyber-Physical Systems

Rüdiger Ehlers, University of Bremen

September 2017

# GR(1) Synthesis (Bloem et al., 2012) – Main idea (1)

**What are we willing to trade?**

...the full expressivity of LTL!

# GR(1) Synthesis (Bloem et al., 2012) – Main idea (1)

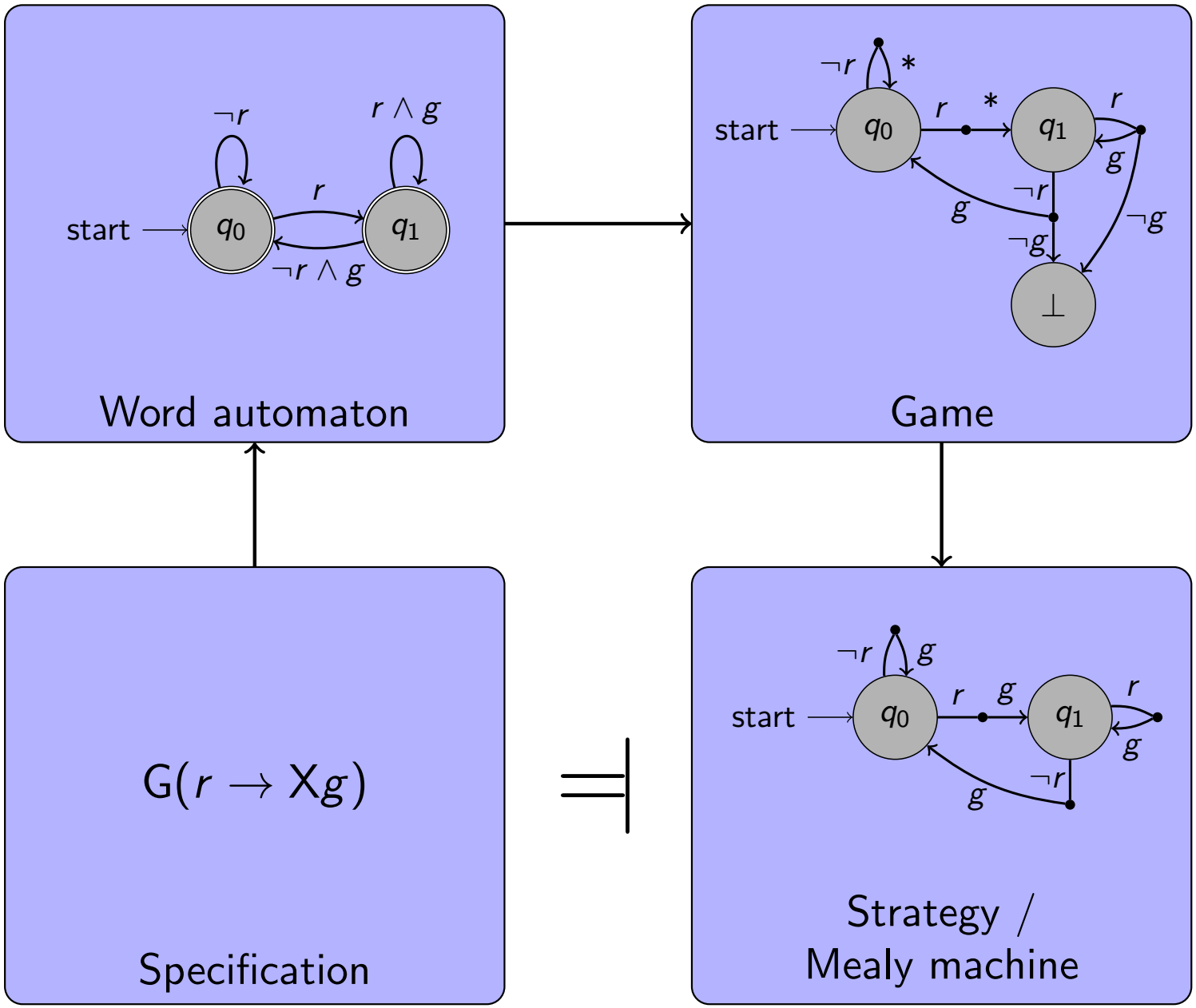
## What are we willing to trade?

...the full expressivity of LTL!

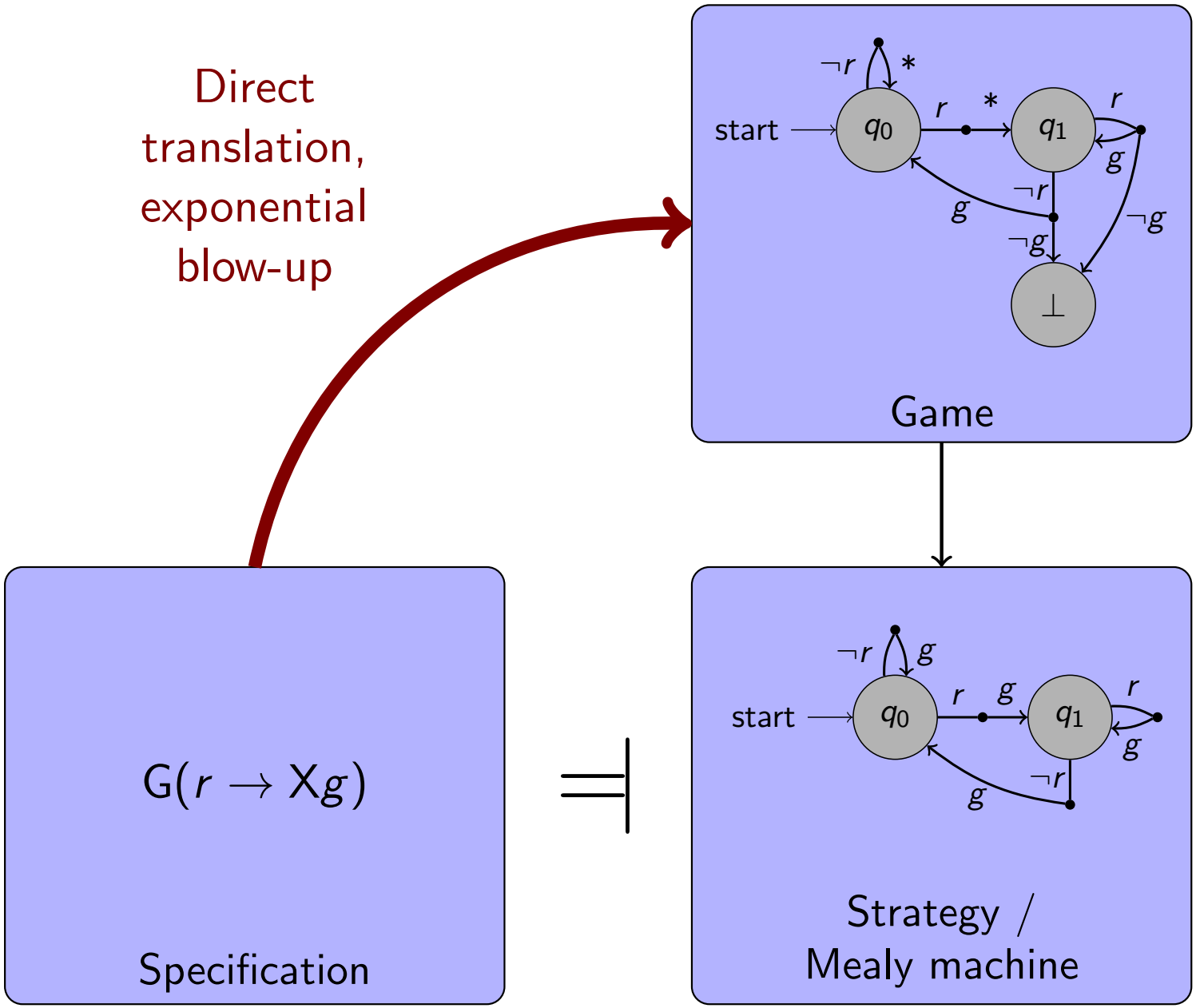
## What do we want?

A reduction in time complexity from doubly-exponential to singly exponential!

# GR(1) Synthesis (Bloem et al., 2012) – Main idea (2)



# GR(1) Synthesis (Bloem et al., 2012) – Main idea (2)



# GR(1) – What should be supported?

## Computation model

We choose a Mealy-type computation model

# GR(1) – What should be supported?

## Computation model

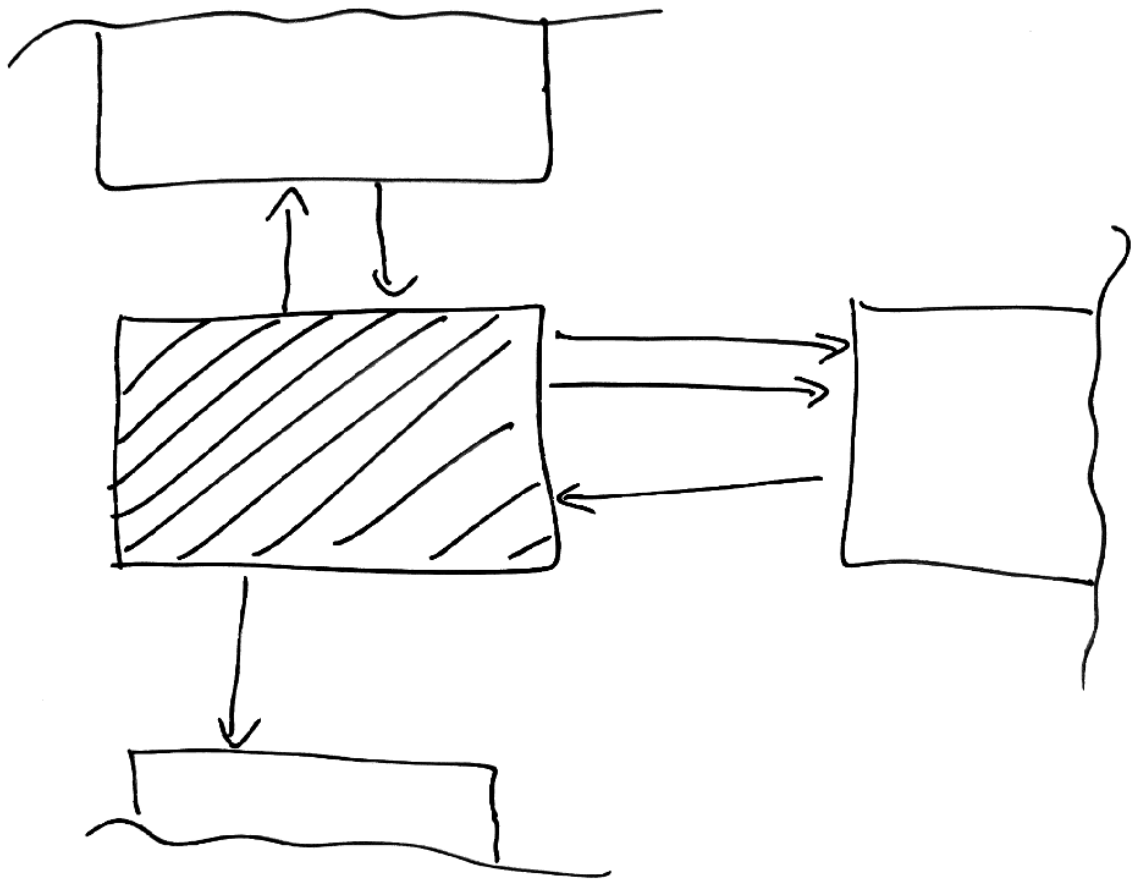
We choose a Mealy-type computation model

## Focus

A specification consists of *assumptions* and *guarantees*, each of which are either

- initialization properties,
- basic safety properies, or
- basic liveness properties.

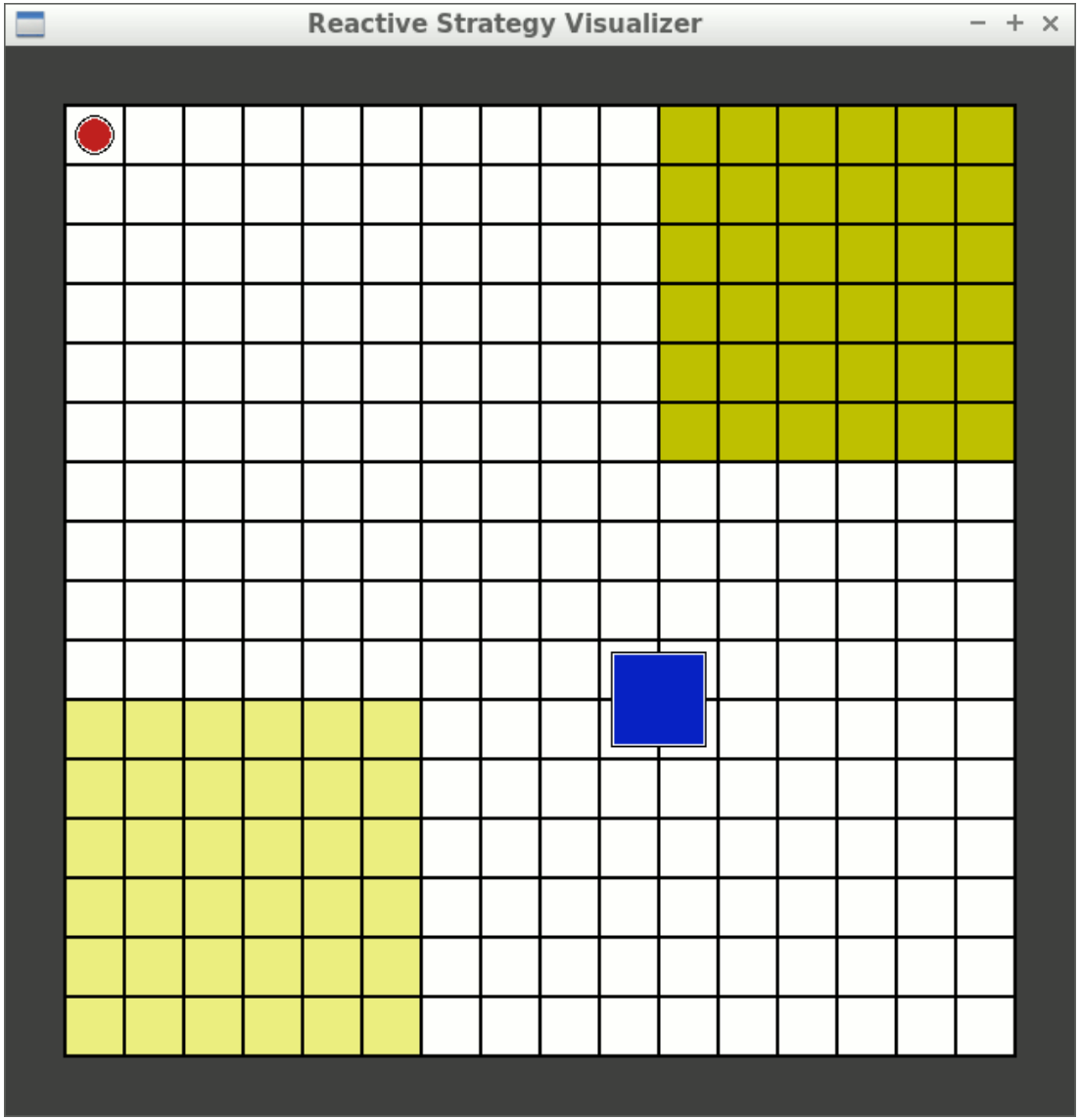
# Assumptions and guarantees in specifications



## Specification shape

$$\left(\bigwedge \text{Assumptions}\right) \rightarrow \left(\bigwedge \text{Guarantees}\right)$$

# Demo – Assumptions & Guarantees



# GR(1) – Overall specification shape

## Specification shape

$$\left(\bigwedge \text{Assumptions}\right) \rightarrow \left(\bigwedge \text{Guarantees}\right)$$

# GR(1) – Overall specification shape

## Specification shape

$$\left(\varphi_i^a \wedge \varphi_s^a \wedge \varphi_l^a\right) \rightarrow \left(\varphi_i^g \wedge \varphi_s^g \wedge \varphi_l^g\right)$$

# GR(1) – Overall specification shape

## Specification shape

$$\left( \underbrace{\varphi_i^a}_{\substack{\text{initialization} \\ \text{assumptions}}} \wedge \underbrace{\varphi_s^a}_{\substack{\text{safety} \\ \text{assumptions}}} \wedge \underbrace{\varphi_l^a}_{\substack{\text{liveness} \\ \text{assumptions}}} \right) \rightarrow (\varphi_i^a \wedge \varphi_s^a \wedge \varphi_l^a)$$

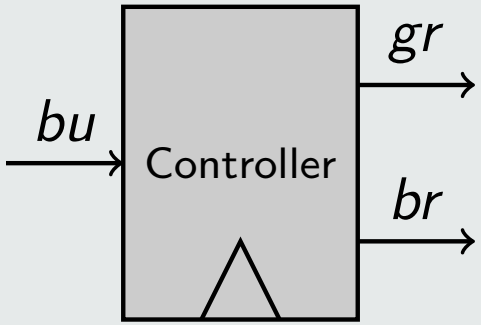
# GR(1) – Overall specification shape

## Specification shape

$$\left(\varphi_i^g \wedge \varphi_s^g \wedge \varphi_l^g\right) \rightarrow \left( \underbrace{\varphi_i^g}_{\substack{\text{initialization} \\ \text{guarantees}}} \wedge \underbrace{\varphi_s^g}_{\substack{\text{safety} \\ \text{guarantees}}} \wedge \underbrace{\varphi_l^g}_{\substack{\text{liveness} \\ \text{guarantees}}} \right)$$

# Specification parts: Initialization assumptions

## Controller shape – Coffee machine example




Here, $\mathrm{AP}_I = \{bu\}$, $\mathrm{AP}_O = \{gr, br\}$

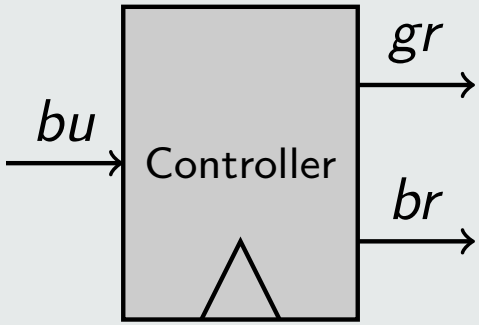
## Initialization assumptions

These are properties without a temporal operator, only over $\mathrm{AP}_I$.
Example:

- $\neg bu$

# Specification parts: Safety assumptions

## Controller shape – Coffee machine example




Here, $\mathrm{AP}_I = \{bu\}$, $\mathrm{AP}_I = \{gr, br\}$

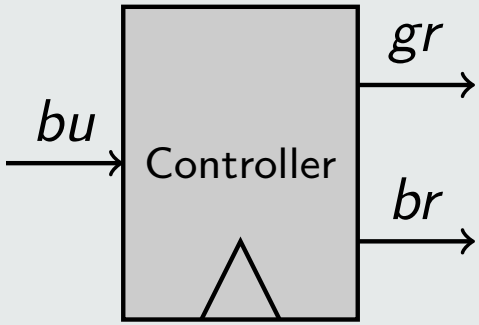
## Safety assumptions

These are properties of the form $\mathbf{G}(\psi)$ where $\psi$ is a Boolean formula over $\mathrm{AP}_I \cup \mathrm{AP}_O \cup \{\mathbf{X}\, y \mid y \in \mathrm{AP}_I\}$. Examples:

- $\mathbf{G}(bu \rightarrow \mathbf{X}\neg bu)$
- $\mathbf{G}((gr \vee br) \rightarrow \mathbf{X}\neg bu)$

# Specification parts: Liveness assumptions

## Controller shape – Coffee machine example




Here, $\mathrm{AP}_I = \{bu\}$, $\mathrm{AP}_I = \{gr, br\}$

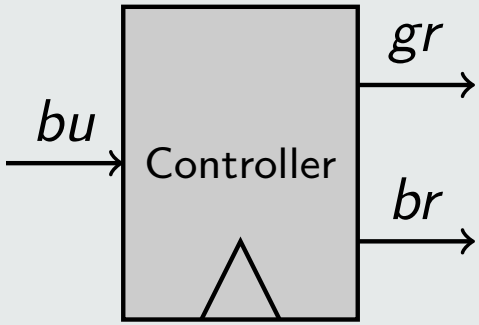
## Liveness assumptions

These are properties of the form $\mathbf{GF}(\psi)$ where $\psi$ is a Boolean formula over $\mathrm{AP}_I \cup \mathrm{AP}_O \cup \{\mathbf{X}\, y \mid y \in \mathrm{AP}_I \cup \mathrm{AP}_O\}$. Examples:

- $\mathbf{GF}(bu)$
- $\mathbf{GF}(\neg br \wedge \neg gr \wedge \mathbf{X} bu)$

# Specification parts: Initialization guarantees

## Controller shape – Coffee machine example




Here, $\mathrm{AP}_I = \{bu\}$, $\mathrm{AP}_I = \{gr, br\}$

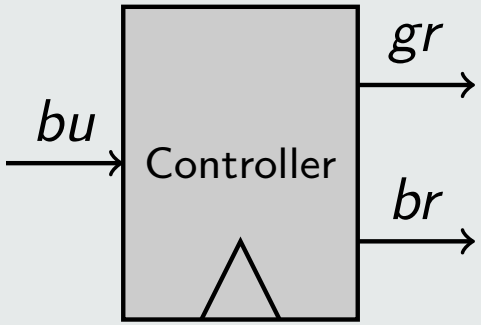
## Initialization guarantees

These are properties without a temporal operator, only over $\mathrm{AP}_I \cup \mathrm{AP}_O$.
Example:

- $\neg gr \wedge \neg br$
- $\neg bu \rightarrow (\neg gr \wedge \neg br)$

# Specification parts: Safety guarantees

## Controller shape – Coffee machine example




Here, $\mathrm{AP}_I = \{bu\}$, $\mathrm{AP}_I = \{gr, br\}$

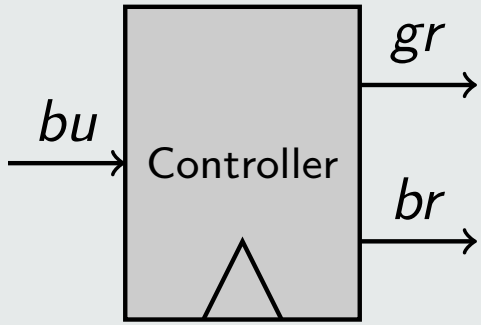
## Safety guarantees

These are properties of the form $\mathbf{G}(\psi)$ where $\psi$ is a Boolean formula over $\mathrm{AP}_I \cup \mathrm{AP}_O \cup \{\mathbf{X}\, y \mid y \in \mathrm{AP}_I \cup \mathrm{AP}_O\}$. Examples:

- $\mathbf{G}(gr \rightarrow \mathbf{X}\neg gr)$
- $\mathbf{G}(gr \wedge \mathbf{X}bu \rightarrow \mathbf{X}gr)$

# Specification parts: Liveness guarantees

## Controller shape – Coffee machine example



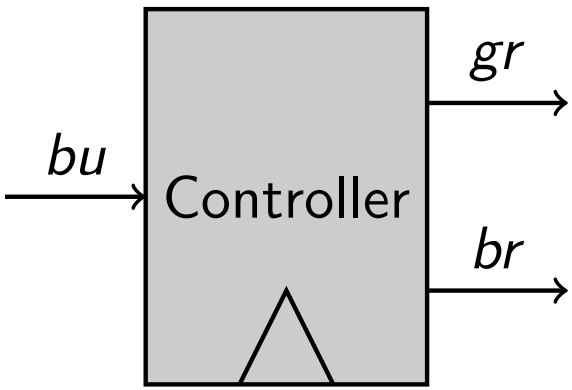

Here, $\mathrm{AP}_I = \{bu\}$, $\mathrm{AP}_I = \{gr, br\}$

## Liveness guarantees

These are properties of the form $\mathbf{GF}(\psi)$ where $\psi$ is a Boolean formula over $\mathrm{AP}_I \cup \mathrm{AP}_O \cup \{\mathbf{X}\, y \mid y \in \mathrm{AP}_I \cup \mathrm{AP}_O\}$. Examples:

- $\mathbf{GF}(gr \wedge \mathbf{X} br)$
- $\mathbf{GF}(bu \vee br)$

# GR(1) (Mealy) execution semantics step-by-step



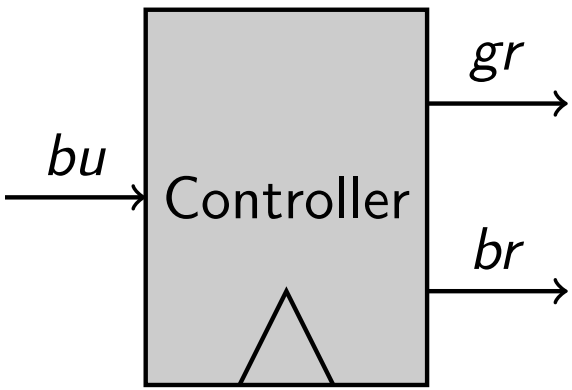

## Atomic propositions

- $\mathrm{AP}_I = \{\mathit{button}\}$
- $\mathrm{AP}_O = \{\mathit{grind}, \mathit{brew}\}$

## A run of the system

$$\rho = \left( \right)$$

# GR(1) (Mealy) execution semantics step-by-step




## Atomic propositions

- $AP_I = \{button\}$
- $AP_O = \{grind, brew\}$

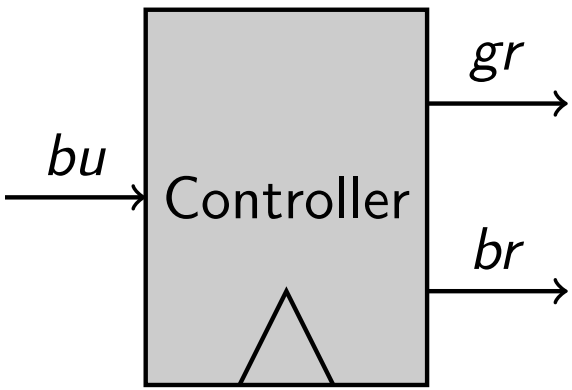
## A run of the system

$$\rho = \begin{pmatrix} 0 \\ \\ \end{pmatrix}$$

## Step 1

The environment selects values for $AP_I$ that satisfy the environment initialization assumptions

# GR(1) (Mealy) execution semantics step-by-step




## Atomic propositions

- $\mathrm{AP}_I = \{\mathit{button}\}$
- $\mathrm{AP}_O = \{\mathit{grind}, \mathit{brew}\}$

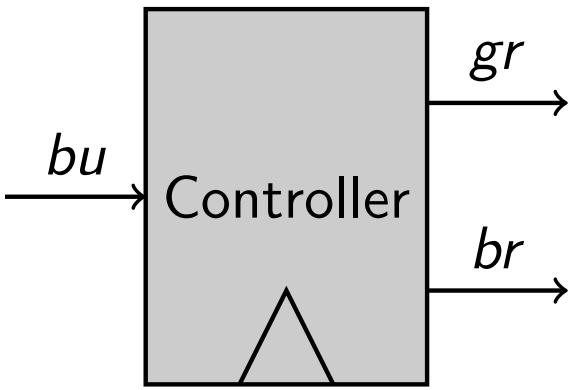
## A run of the system

$$\rho = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

## Step 2

The system selects values for $\mathrm{AP}_O$ such that the first element of $\rho$ satisfies all initialization guarantees

# GR(1) (Mealy) execution semantics step-by-step




## Atomic propositions

- $\mathrm{AP}_I = \{\mathit{button}\}$
- $\mathrm{AP}_O = \{\mathit{grind}, \mathit{brew}\}$

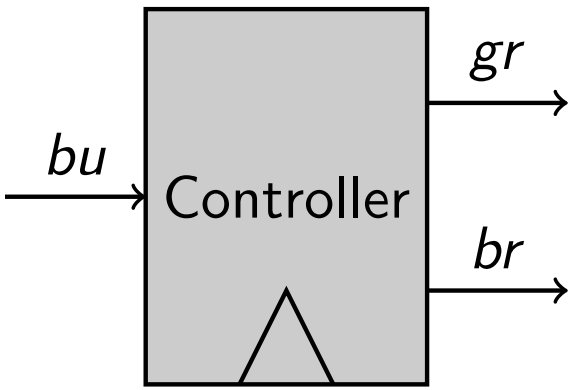
## A run of the system

$$\rho = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ \\ \end{pmatrix}$$

## Step $2 \cdot i + 1$

The environment selects values for $\mathrm{AP}_I$ such that the last element of $\rho$ and the new values for $\mathrm{AP}_I$ satisfy the environment safety assumptions

# GR(1) (Mealy) execution semantics step-by-step




## Atomic propositions

- $\mathrm{AP}_I = \{\mathit{button}\}$
- $\mathrm{AP}_O = \{\mathit{grind}, \mathit{brew}\}$

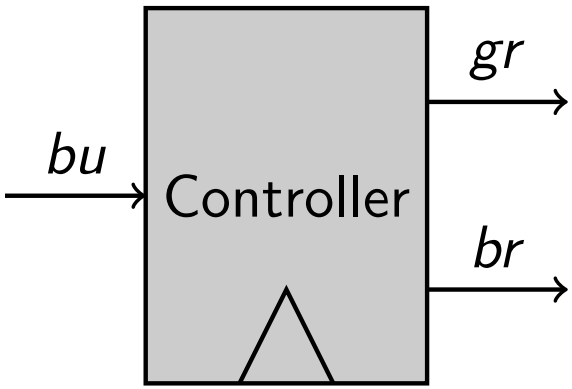
## A run of the system

$$\rho = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

## Step $2 \cdot i + 2$

The system selects values for $\mathrm{AP}_O$ such that the last element of $\rho$ and the new values for $\mathrm{AP}_I$ and $\mathrm{AP}_O$ satisfy the system safety guarantees

# GR(1) (Mealy) execution semantics step-by-step




## Atomic propositions

- $\mathrm{AP}_I = \{\mathit{button}\}$
- $\mathrm{AP}_O = \{\mathit{grind}, \mathit{brew}\}$

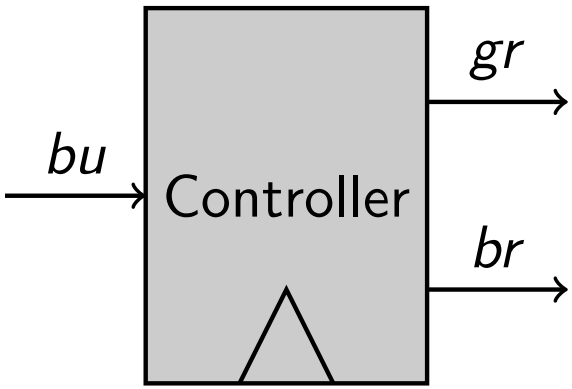
## A run of the system

$$\rho = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ \\ \end{pmatrix}$$

## Step $2 \cdot i + 1$

The environment selects values for $\mathrm{AP}_I$ such that the last element of $\rho$ and the new values for $\mathrm{AP}_I$ satisfy the environment safety assumptions

# GR(1) (Mealy) execution semantics step-by-step




## Atomic propositions

- $\mathrm{AP}_I = \{\mathit{button}\}$
- $\mathrm{AP}_O = \{\mathit{grind}, \mathit{brew}\}$

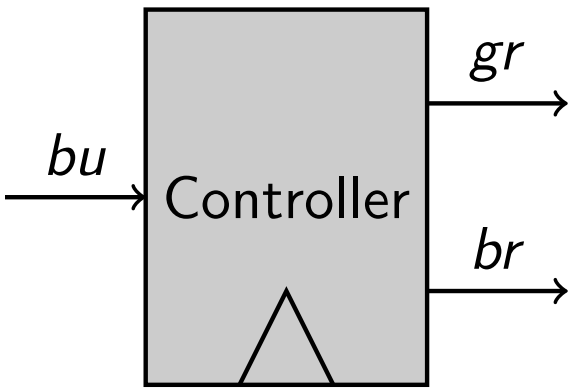
## A run of the system

$$\rho = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

## Step $2 \cdot i + 2$

The system selects values for $\mathrm{AP}_O$ such that the last element of $\rho$ and the new values for $\mathrm{AP}_I$ and $\mathrm{AP}_O$ satisfy the system safety guarantees

# GR(1) (Mealy) execution semantics step-by-step




## Atomic propositions

- $\mathrm{AP}_I = \{\mathit{button}\}$
- $\mathrm{AP}_O = \{\mathit{grind}, \mathit{brew}\}$

## A run of the system

$$\rho = \begin{pmatrix}0\\0\\0\end{pmatrix}\begin{pmatrix}1\\1\\0\end{pmatrix}\begin{pmatrix}0\\1\\0\end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix}\ldots$$

## And so on...

This process continues ad infinitum.

# GR(1) Semantics – Who wins the game?

## Finitary winning

If at some point, one of the players does not stick to the rules of the game, then the player doing so first **loses** the game.

## Otherwise: Infinitary winning

If both players play according to the rules, then the system player wins if and only if the winning condition

$$\varphi_I^a \rightarrow \varphi_I^g$$

is fulfilled.

# Let's explore the semantics by example (1)

## GR(1) synthesis tool used

`slugs` – Live web-based version available at
`http://webslugs.ruediger-ehlers.de`

## Specification

```
[INPUT]
bu

[OUTPUT]
br
gr

[ENV_INIT]

[SYS_INIT]
gr <-> bu
! br
```

# Let's explore the semantics by example (2)

## Added specification parts

```
[SYS_TRANS]
br' <-> gr
gr' -> bu'

[ENV_TRANS]
bu' -> !gr & !br
```

# Let's explore the semantics by example (2)

## Added specification parts

```
[SYS_TRANS]
br' <-> gr
gr' -> bu'

[ENV_TRANS]
bu' -> !gr & !br
```

## Observation

The system **can** now make coffee, but does not *have* to.

# Let's explore the semantics by example (3)

## Added specification parts

```
[SYS_LIVENESS]
br
```

# Let's explore the semantics by example (3)

## Added specification parts

```
[SYS_LIVENESS]
br
```

## Observation

Since the system cannot enforce a button press, it now loses

# Let's explore the semantics by example (4)

## Added specification parts

```
[ENV_LIVENESS]
bu
```

# Let's explore the semantics by example (4)

## Added specification parts

```
[ENV_LIVENESS]
bu
```

# Let's explore the semantics by example (4)

## Added specification parts

```
[ENV_LIVENESS]
bu
```

## Observation

Now everything works as expected!

# Beware the semantics of GR(1) – Part I

## Note

There is a discrepancy between the presentation of a GR(1) problem in the form

$$\left(\bigwedge \text{Assumptions}\right) \rightarrow \left(\bigwedge \text{Guarantees}\right)$$

and the step-by-step execution explained above.

# Beware the semantics of GR(1) – Part I

## Note

There is a discrepancy between the presentation of a GR(1) problem in the form

$$\left(\bigwedge \text{Assumptions}\right) \rightarrow \left(\bigwedge \text{Guarantees}\right)$$

and the step-by-step execution explained above.

## Example (1)

$$(\mathbf{GF}r \wedge \mathbf{G}\neg r) \rightarrow (\mathbf{G}g \wedge \mathbf{G}\neg g)$$

# Beware the semantics of GR(1) – Part II

## Note

There is a discrepancy between the presentation of a GR(1) problem in the form

$$\left(\bigwedge \text{Assumptions}\right) \rightarrow \left(\bigwedge \text{Guarantees}\right)$$

and the step-by-step execution explained above.

## Example (2)

$$(\mathbf{G}r \wedge \mathbf{G}\neg r) \rightarrow (\mathbf{GX}g \wedge \mathbf{GX}\neg g)$$

# Syntactic Extension to GR(1) – Counters

## Using counters

To simplify working with cyber-physical systems, we will syntactically extend the set of GR(1) specifications by *counter variables*, which are actually binary-encoded into the atomic propositions.

## Note

In LTL, this does not make sense:

$$\mathbf{G}(\mathit{counter} \leq \mathbf{X}(\mathit{counter}) + 7)$$

But some synthesis tools such as `slugs` and `TuLiP` (Wongpiromsarn et al., 2011) support this anyway.

# Syntactic Extension to GR(1) – Counters in Slugs

## Version with counters

```
[INPUT]
a:0...15

[OUTPUT]
b

[ENV_INIT]
a >= 3

[ENV_TRANS]
a' <= a + 8

...
```

## Version without counters

```
[INPUT]
a@0.0.15
a@1
a@2
a@3

[OUTPUT]
b

[ENV_INIT]
a@0.0.15 & a@1 | a@2 | a@3

[ENV_TRANS]
...
```

# Slugs – Example with counters

```
[INPUT]
u

[OUTPUT]
c:0...10

[SYS_INIT]
c = 0

[SYS_TRANS]
u' -> c'=c+1
```

# How GR(1) synthesis works

# Step 1: Building a synthesis game

## Basic idea

The state space of the game is $2^{\mathsf{AP}_I \cup \mathsf{AP}_O}$.

- The initialization assumptions and guarantees are used to define the set of initial states of the game.
- The safety assumptions and guarantees are used to define the transition structure of the game
- The liveness assumptions and guarantees are used to define the winning condition of the game.

# Step 1: Building a synthesis game

## Basic idea

The state space of the game is $2^{AP_I \cup AP_O}$.

- The initialization assumptions and guarantees are used to define the set of initial states of the game.
- The safety assumptions and guarantees are used to define the transition structure of the game
- The liveness assumptions and guarantees are used to define the winning condition of the game.

## Central property of the game

$\rightarrow$ size is exponential in $|AP_I \cup AP_O|$.

# Example

## Specification

$$(\mathbf{GF}x \wedge \mathbf{G}(\neg x \vee \neg \mathbf{X}\, x)) \rightarrow (\mathbf{G}((\neg x \wedge y) \rightarrow \mathbf{X}\, x) \wedge \mathbf{GFX}y \wedge (x \leftrightarrow y))$$

# Example

## Specification

$$(\mathbf{GF}x \wedge \mathbf{G}(\neg x \vee \neg \mathbf{X}\, x)) \rightarrow (\mathbf{G}((\neg x \wedge y) \rightarrow \mathbf{X}\, x) \wedge \mathbf{GFX}y \wedge (x \leftrightarrow y))$$

## Breaking the specification into pieces

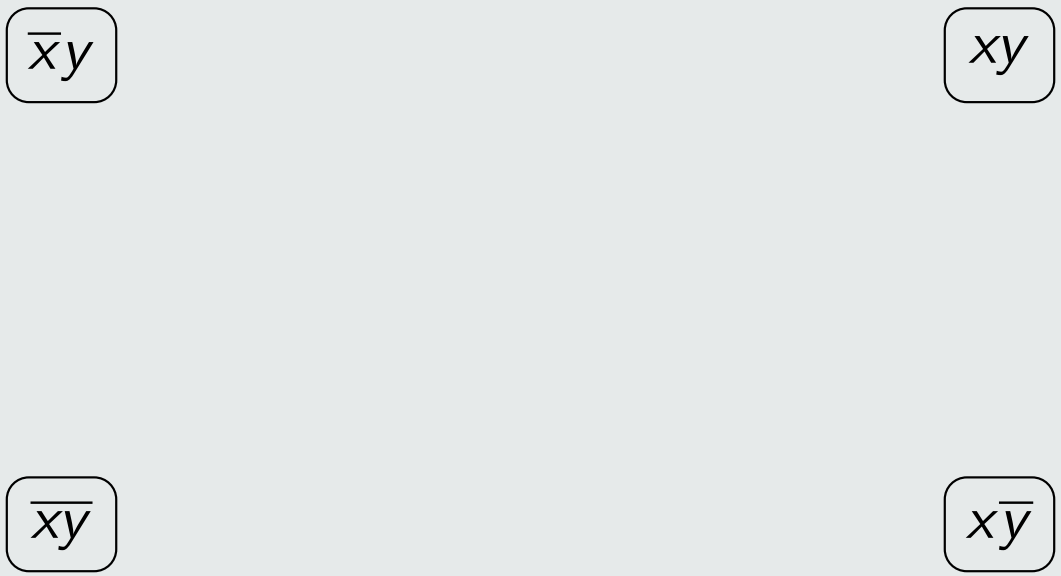
- Initialization assumptions: none
- Safety assumptions: $\mathbf{G}(\neg x \vee \neg \mathbf{X}\, x)$
- Liveness assumptions: $\mathbf{GF}x$
- Initialization guarantees: $(x \leftrightarrow y)$
- Safety guarantees: $\mathbf{G}((\neg x \wedge y) \rightarrow \mathbf{X}\, x)$
- Liveness guarantees: $\mathbf{GFX}y$

# Building the game

## Relevant specification parts for building the game

- Safety assumptions: $\mathbf{G}(\neg x \vee \neg \mathbf{X}\, x)$
- Safety guarantees: $\mathbf{G}((\neg x \wedge y) \rightarrow \mathbf{X}\, x)$

## Game

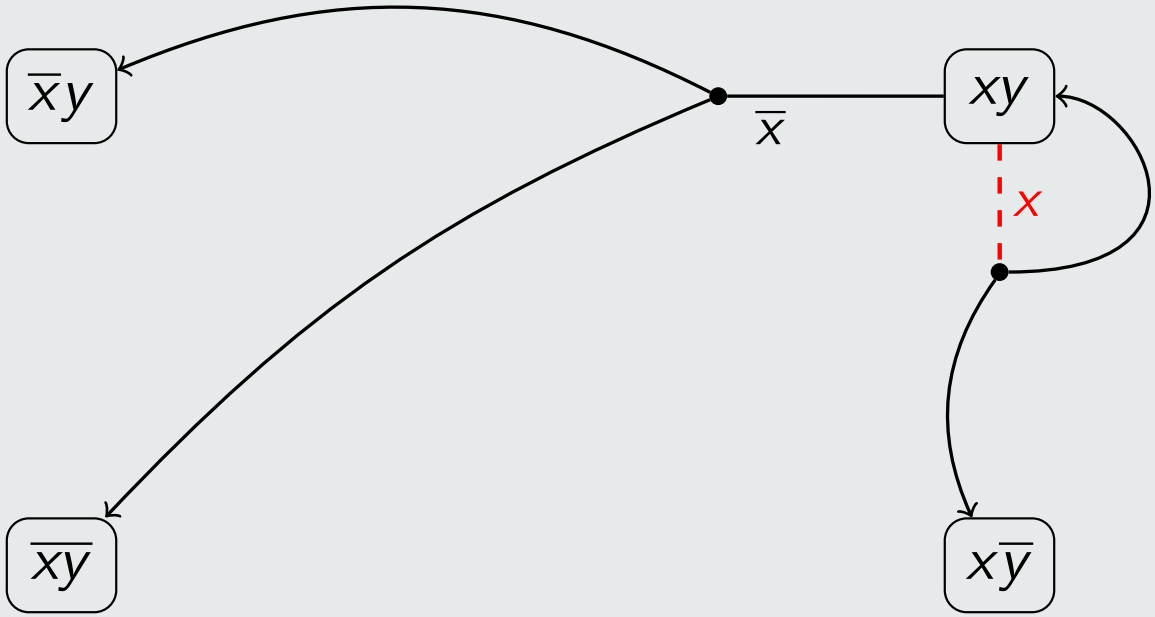


# Building the game

## Relevant specification parts for building the game

- Safety assumptions: $\mathbf{G}(\neg x \vee \neg \mathbf{X}\, x)$
- Safety guarantees: $\mathbf{G}((\neg x \wedge y) \rightarrow \mathbf{X}\, x)$

## Game

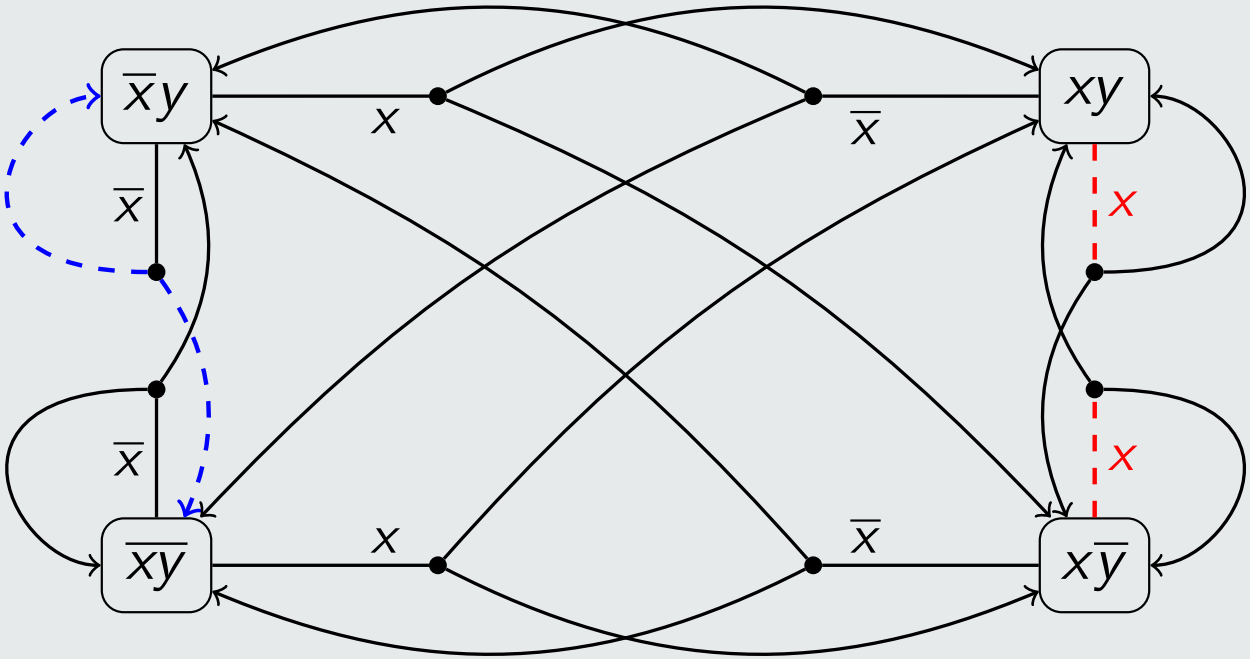


# Building the game

## Relevant specification parts for building the game

- Safety assumptions: $\mathbf{G}(\neg x \vee \neg \mathbf{X}\, x)$
- Safety guarantees: $\mathbf{G}((\neg x \wedge y) \rightarrow \mathbf{X}\, x)$

## Game



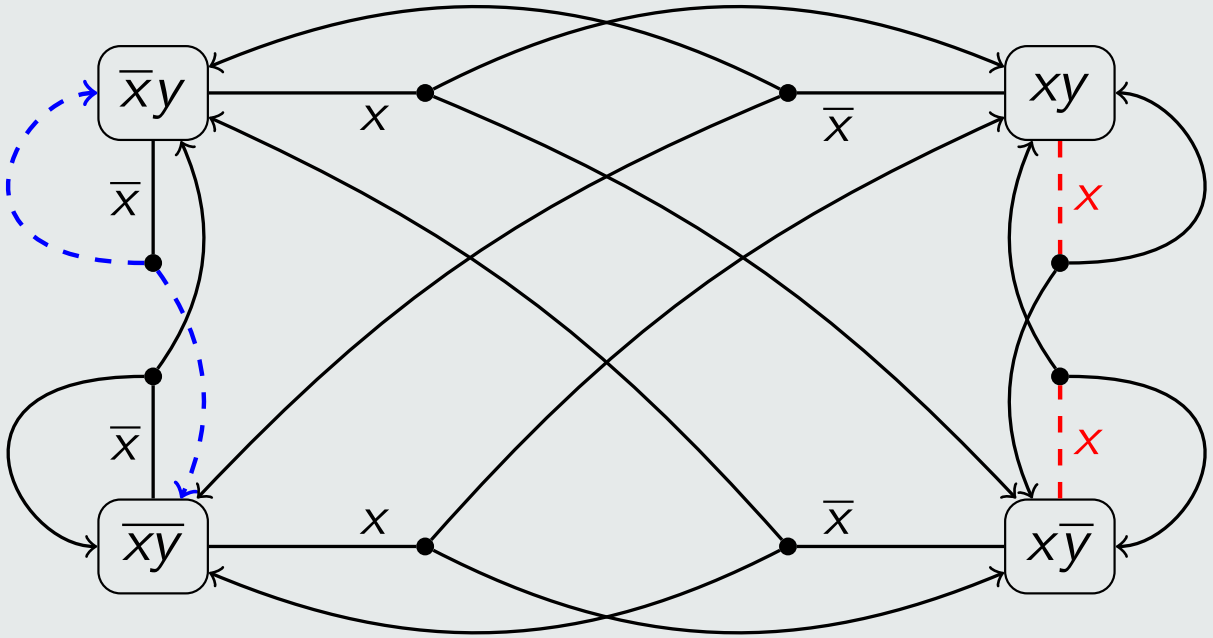
# Solving GR(1) games – Step 1: Safety game solving

## Process

Compute the largest set of (winning) game states $W$ such that:

- a state is removed from $W$ if the environment has a (legal) successor state such that all successors are non-winning (or the transition is illegal)

## Game



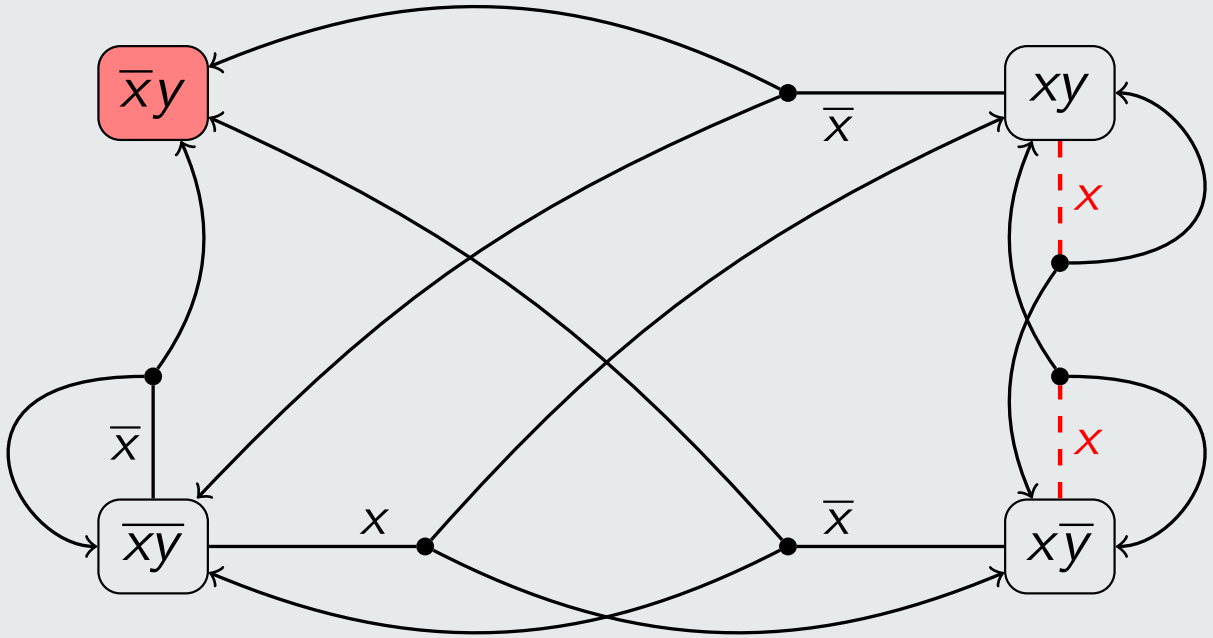
# Solving GR(1) games – Step 1: Safety game solving

## Process

Compute the largest set of (winning) game states $W$ such that:

- a state is removed from $W$ if the environment has a (legal) successor state such that all successors are non-winning (or the transition is illegal)

## Game



# Solving GR(1) games – Step 1: Safety game solving

## Process

Compute the largest set of (winning) game states $W$ such that:

- a state is removed from $W$ if the environment has a (legal) successor state such that all successors are non-winning (or the transition is illegal)

## Towards modelling this as a $\mu$-calculus formula

We search for the largest $W \subseteq 2^{\mathsf{AP}_I \cup \mathsf{AP}_O}$ such that:

$$W = \mathsf{EnfPre}(W),$$

where for every state set $X \subseteq 2^{\mathsf{AP}_I \cup \mathsf{AP}_O}$, we have that $\mathsf{EnfPre}(X)$ contains all $x \in 2^{\mathsf{AP}_I \cup \mathsf{AP}_O}$ such that the system player can enforce that after one move of each player, the play is in a state in $X$.

# Solving GR(1) games – Step 1: Safety game solving

## Towards modelling this as a $\mu$-calculus formula

To obtain set $W$, we can compute (for finite-sized games):

$$W_0 = 2^{\mathrm{AP}_I \cup \mathrm{AP}_O}$$

followed by

$$W_1 = \mathrm{EnfPre}(W_0),$$

$$W_2 = \mathrm{EnfPre}(W_1)$$

and so on, until we reach a *fixpoint*.

# Solving GR(1) games – Step 1: Safety game solving

## Towards modelling this as a $\mu$-calculus formula

To obtain set $W$, we can compute (for finite-sized games):

$$W_0 = 2^{\mathrm{AP}_I \cup \mathrm{AP}_O}$$

followed by

$$W_1 = \mathrm{EnfPre}(W_0),$$

$$W_2 = \mathrm{EnfPre}(W_1)$$

and so on, until we reach a *fixpoint*.

## Modelling as a $\mu$-calculus formula

$$W = \nu X.\mathrm{EnfPre}(X)$$

# Solving GR(1) games – Step 2: Reachability game solving

## The next step

Now we need to take the winning condition of the GR(1) game into consideration. For our example GR(1) game, this is:

$$(\mathbf{GF}x) \rightarrow (\mathbf{GFX}y)$$

# Solving GR(1) games – Step 2: Reachability game solving

## The next step

Now we need to take the winning condition of the GR(1) game into consideration. For our example GR(1) game, this is:

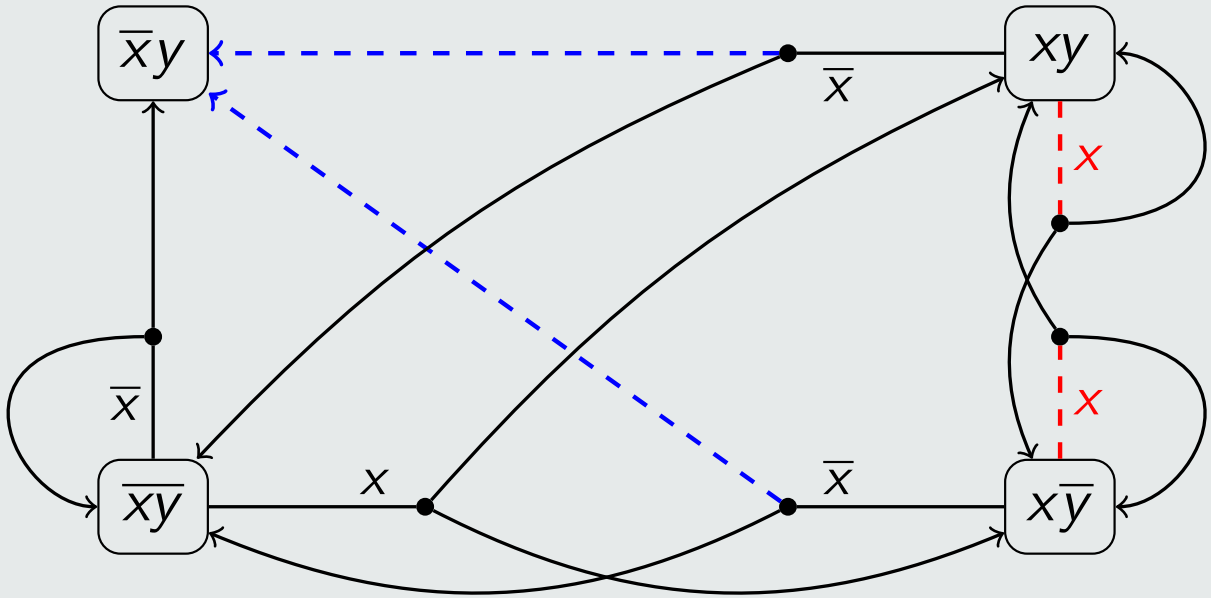
$$(\mathbf{GF}x) \rightarrow (\mathbf{GFX}y)$$

## Coming up

Let us have a look at from which states in the game the system player can enforce that eventually a transition is taken along which $\mathbf{X}y$ is satisfied.

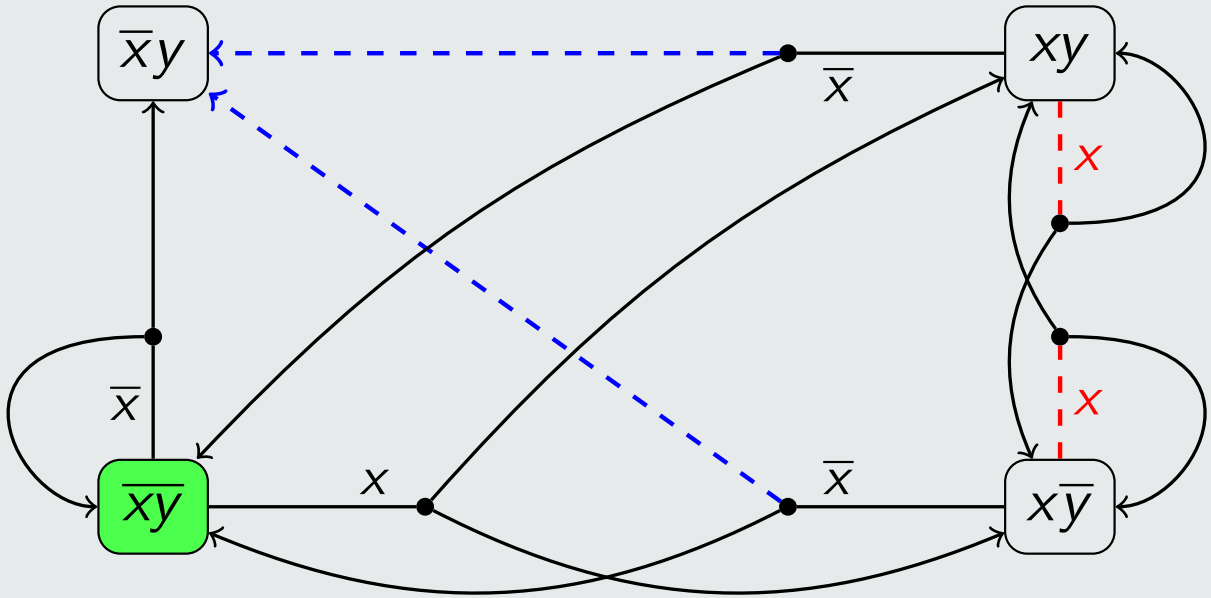
# Eventually taking a transition satisfying $\mathbf{X}y$

## The game (modified!)



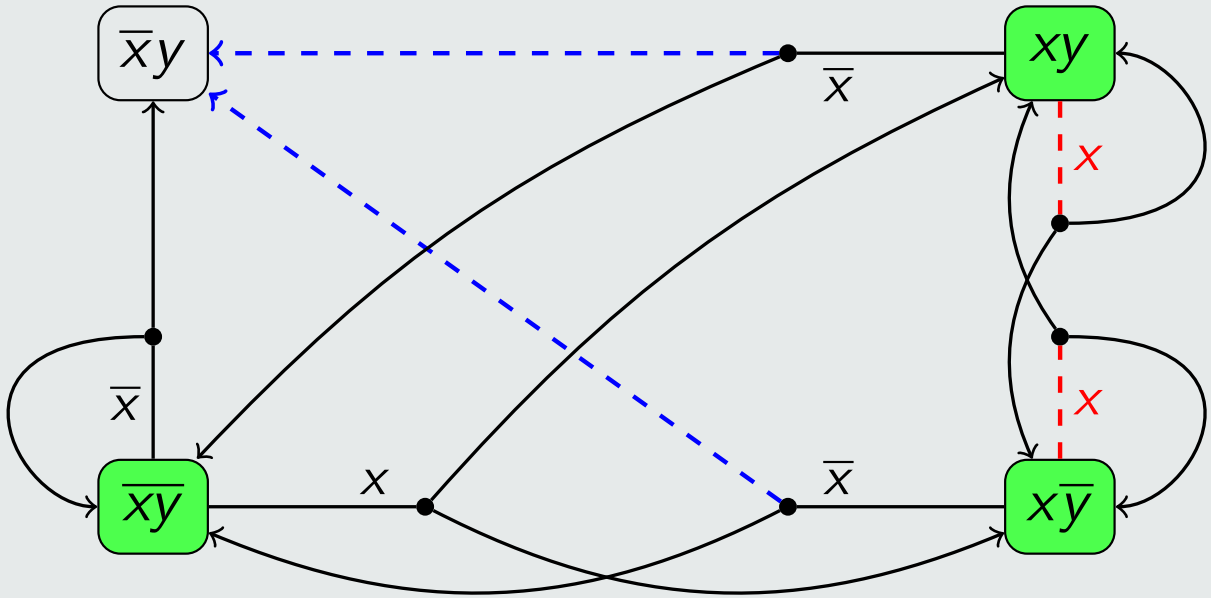
# Eventually taking a transition satisfying $\mathbf{X}y$

## The game (modified!)



# Eventually taking a transition satisfying $\mathbf{X}y$

## The game (modified!)



# Solving GR(1) games – Step 2: Reachability game solving

## New $\mu$-calculus equation for eventually taking goal transition $\psi$

$$\mu X.X \cup \mathsf{EnfPre}(X' \cup \psi)$$

...using the extension of EnfPre to range over transition instead of states, where a dash indicates a state reached after a transition.

# Solving GR(1) games – Step 3: Environment goals

## Next step

Now the system only needs to reach the next goal under the assumption that the environment fulfils its liveness assumptions:

$$(\mathbf{GF}x) \rightarrow (\mathbf{FX}y)$$

# Solving GR(1) games – Step 3: Environment goals

## Next step

Now the system only needs to reach the next goal under the assumption that the environment fulfils its liveness assumptions:

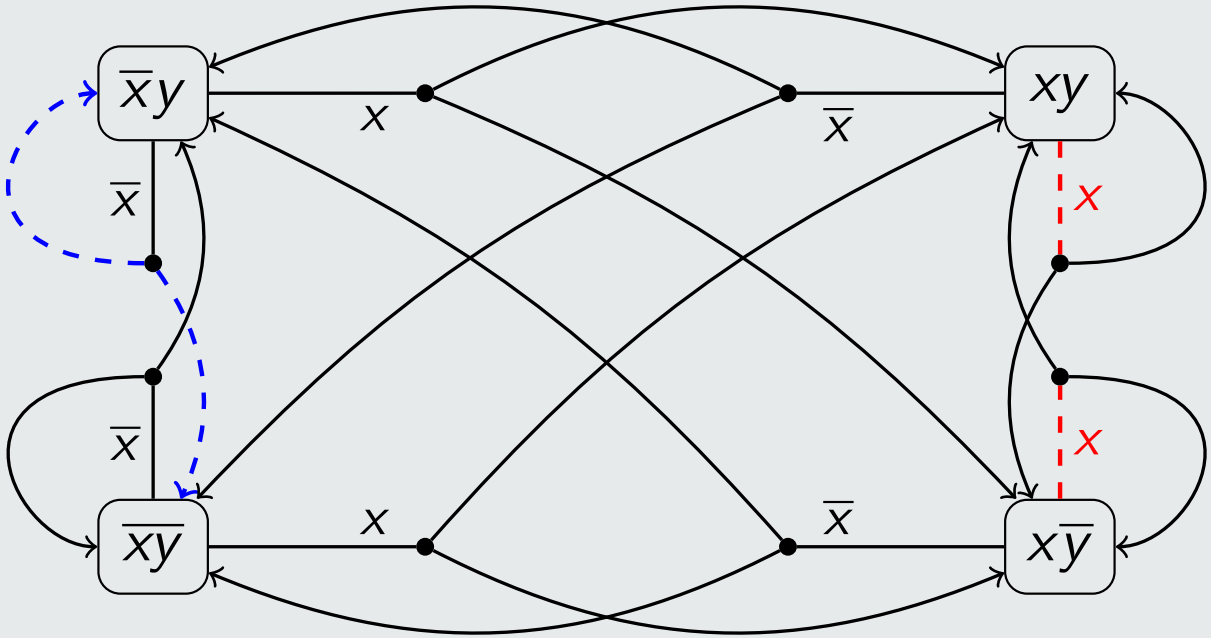
$$(\mathbf{GF}x) \rightarrow (\mathbf{FX}y)$$

## Idea

The system now only needs to make progress towards its *goal* whenever the environment reaches one of its *goals*.

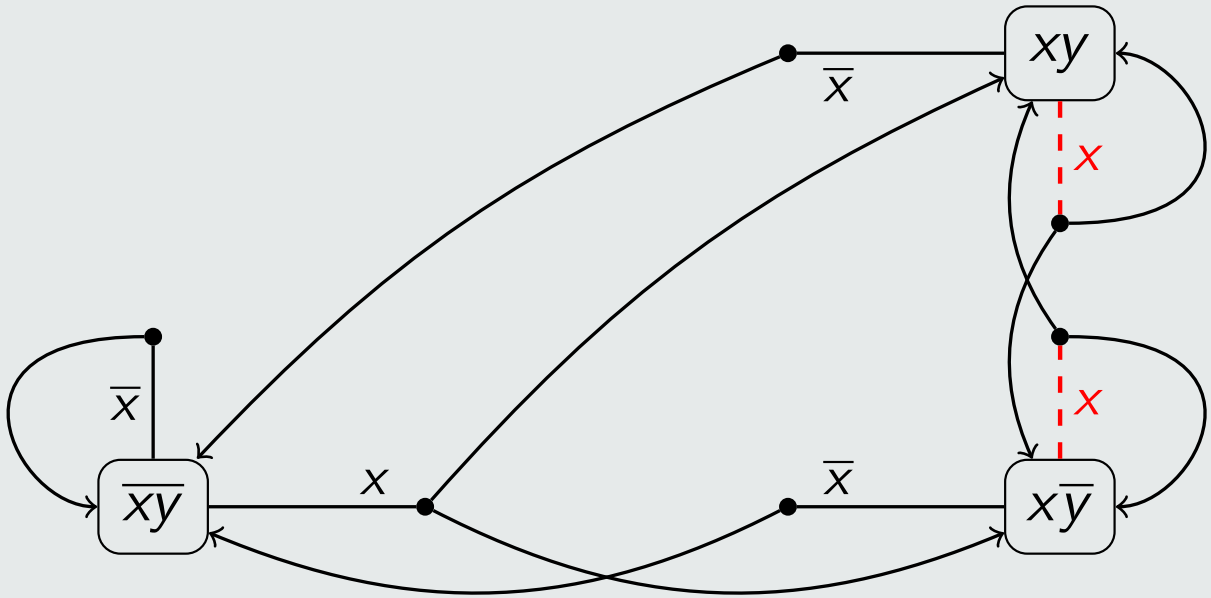
# Working on $(\mathbf{GF}x) \rightarrow (\mathbf{FX}y)$

## The game (with the losing states)



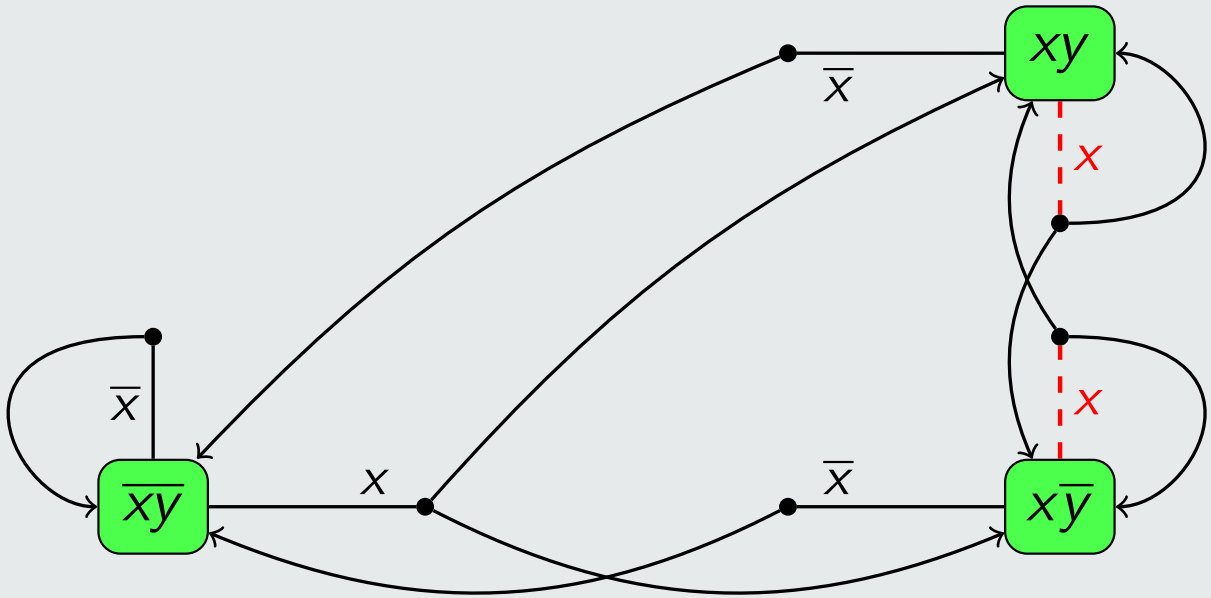
# Working on $(\mathbf{GF}x) \rightarrow (\mathbf{FX}y)$

## The game (without the losing states)



# Working on $(\mathbf{GF}x) \rightarrow (\mathbf{FX}y)$

## The game (without the losing states)



# Solving GR(1) games – Step 3: Environment goals

## New $\mu$-calculus formula, first step

Idea: In every step of the system's strategy execution, the strategy either (1) waits for the environment to reach a goal or (2) moves closer towards its own goal:

$$\mu Y.\mathrm{EnfPre}(\psi^g \cup Y') \cup \nu X.\mathrm{EnfPre}((X' \cap \neg\psi^a) \cup Y')$$

# Solving GR(1) games – Step 3: Environment goals

## New $\mu$-calculus formula, first step

Idea: In every step of the system's strategy execution, the strategy either (1) waits for the environment to reach a goal or (2) moves closer towards its own goal:

$$\mu Y.\text{EnfPre}(\psi^g \cup Y') \cup \nu X.\text{EnfPre}((X' \cap \neg\psi^a) \cup Y')$$

## Small problem

Which of the two cases above holds may not be under the control of the system. Alternative formula:

$$\mu Y.\nu X.\text{EnfPre}(\psi^g \cup Y' \cup (X' \cap \neg\psi^a))$$

# GR(1) Synthesis – Plugging things together

## What is still missing

- The system goals need to be reached infinitely often
- Support for multiple environment goals and system goals

## Completion of the formula

$$\mu Y.\nu X.\text{EnfPre}(\psi^g \cup Y' \cup (X' \cap \neg\psi^a))$$

# GR(1) Synthesis – Plugging things together

## What is still missing

- The system goals need to be reached infinitely often
- Support for multiple environment goals and system goals

## Completion of the formula

$$\mu Y.\nu X.\text{EnfPre}(\psi^g \cup Y' \cup (X' \cap \neg\psi^a))$$

$$\Downarrow$$

$$\nu Z.\mu Y.\nu X.\text{EnfPre}(Z' \cap \psi^g \cup Y' \cup (X' \cap \neg\psi^a))$$

# GR(1) Synthesis – Plugging things together

## What is still missing

- The system goals need to be reached infinitely often
- Support for multiple environment goals and system goals

## Completion of the formula

$$\mu Y.\nu X.\text{EnfPre}(\psi^g \cup Y' \cup (X' \cap \neg\psi^a))$$

$$\Downarrow$$

$$\nu Z.\mu Y.\nu X.\text{EnfPre}(Z' \cap \psi^g \cup Y' \cup (X' \cap \neg\psi^a))$$

$$\Downarrow$$

$$\nu Z.\bigcap_{i=1}^{n} \mu Y.\bigcup_{j=1}^{m} \nu X.\text{EnfPre}(Z' \cap \psi_i^g \cup Y' \cup (X' \cap \neg\psi_j^a))$$

# The final GR(1) fixpoint

$$\nu Z. \bigcap_{j \in \{1,\ldots,n\}} \mu Y. \bigcup_{i \in \{1,\ldots,m\}} \nu X. \mathsf{EnfPre}\big( (Z' \cap \psi_j^g) \cup Y' \cup (\neg\psi_i^a \cap X') \big)$$

# The final GR(1) fixpoint

$$\nu Z. \bigcap_{j\in\{1,\ldots,n\}} \mu Y. \bigcup_{i\in\{1,\ldots,m\}} \nu X. \mathsf{EnfPre}\big( (Z' \cap \psi_j^g) \cup Y' \cup (\neg\psi_i^a \cap X') \big)$$

Reaching the next system goal

# The final GR(1) fixpoint

$$\nu Z. \bigcap_{j\in\{1,\ldots,n\}} \mu Y. \bigcup_{i\in\{1,\ldots,m\}} \nu X. \mathsf{EnfPre}\big( (Z' \cap \psi_j^g) \cup Y' \cup (\neg\psi_i^a \cap X') \big)$$

Reaching the next system goal

Getting closer to the next system goal

# The final GR(1) fixpoint

$$\nu Z. \bigcap_{j \in \{1,\ldots,n\}} \mu Y. \bigcup_{i \in \{1,\ldots,m\}} \nu X. \mathsf{EnfPre}\big( (Z' \cap \psi_j^g) \cup Y' \cup (\neg \psi_i^a \cap X') \big)$$

Reaching the next system goal

Getting closer to the next system goal

Waiting for some environment goal

# The final GR(1) fixpoint

$$\nu Z. \bigcap_{j\in\{1,\ldots,n\}} \mu Y. \bigcup_{i\in\{1,\ldots,m\}} \nu X. \mathrm{EnfPre}\big((Z' \cap \psi_j^g) \cup Y' \cup (\neg\psi_i^a \cap X')\big)$$

Give semantics to $X$, $Y$, and $Z$

Reaching the next system goal

Getting closer to the next system goal

Waiting for some environment goal

# How do the synthesized strategies look like?

## The equation

$$\nu Z. \bigcap_{i=1}^{n} \mu Y. \bigcup_{j=1}^{m} \nu X. \mathsf{EnfPre}(Z' \cap \psi_i^g \cup Y' \cup (X' \cap \neg \psi_j^a))$$

# How do the synthesized strategies look like?

## The equation

$$\nu Z. \bigcap_{i=1}^{n} \mu Y. \bigcup_{j=1}^{m} \nu X. \mathsf{EnfPre}(Z' \cap \psi_i^g \cup Y' \cup (X' \cap \neg \psi_j^a))$$

## Strategy extraction

For every liveness guarantee no. $i \in \{1, \ldots, n\}$, the transitions computed during the computation of the $\nu Y$ prefix points while $Z$ and $X$ are fully evaluated represent the set of transitions getting closer to system goal $i$.

# How do the synthesized strategies look like?

## The equation

$$\nu Z. \bigcap_{i=1}^{n} \mu Y. \bigcup_{j=1}^{m} \nu X. \mathsf{EnfPre}(Z' \cap \psi_i^g \cup Y' \cup (X' \cap \neg \psi_j^a))$$

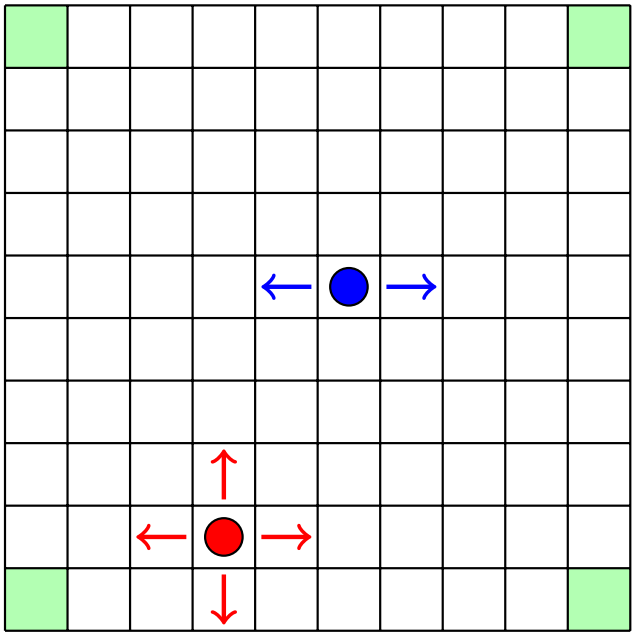
## Strategy extraction

For every liveness guarantee no. $i \in \{1, \ldots, n\}$, the transitions computed during the computation of the $\nu Y$ prefix points while $Z$ and $X$ are fully evaluated represent the set of transitions getting closer to system goal $i$.

## So the final strategy...

...performs the tasks in a round-robin fashion.

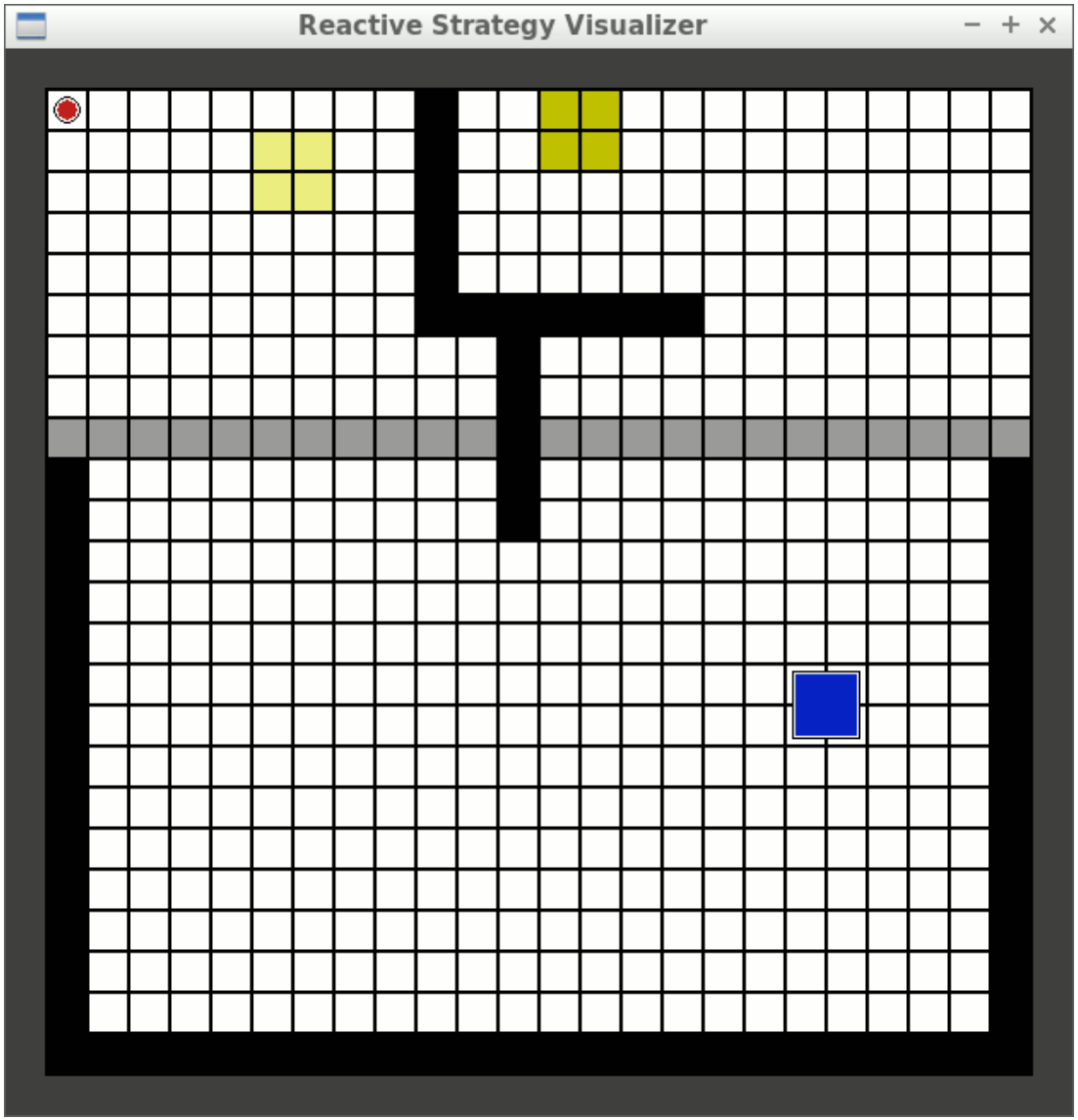
# Toggling through the goals – A simple CPS example



```
[INPUT]
o:0...10

[OUTPUT]
x:0...10
y:0...10

[SYS_INIT]
x=0
y=0
```

```
[ENV_INIT]
o=0

[SYS_TRANS]
x'=x | y'=y
y'<=y+1
y'+1>=y
x'<=x+1
x'+1>=x

[SYS_LIVENESS]
x'=0 & y'=0
x'=10 & y'=0
x'=10 & y'=10
x'=0 & y'=10

[SYS_TRANS]
y'!=5 | o'!=y' & o'+1!=y' & o'!=y'+1

[ENV_LIVENESS]
o'=0
o'=5
o'=10

[ENV_TRANS]
o'<=o+1
o'+1>=o
```

# Another CPS example with a discrete abstraction



# Some general notes on the practice of GR(1) synthesis

## Notes

- Most GR(1) synthesis tools do not allow **X** in the liveness assumptions and guarantees
  $\rightarrow$ No big deal, we can use additional helper variables

# Some general notes on the practice of GR(1) synthesis

## Notes

- Most GR(1) synthesis tools do not allow **X** in the liveness assumptions and guarantees
  $\rightarrow$ No big deal, we can use additional helper variables

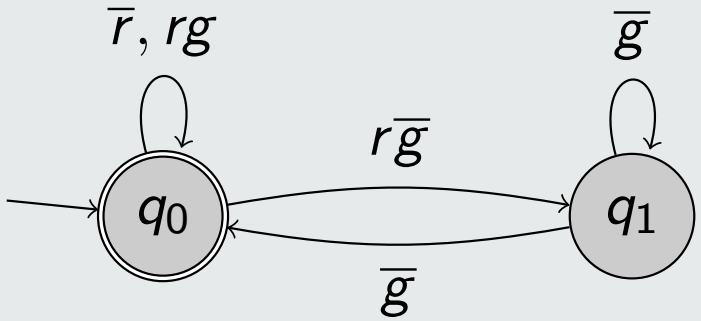
## Example

$$\mathbf{GF}(y \wedge \mathbf{X}y)$$
$$\Downarrow$$
$$\mathbf{GF}(y \wedge p) \wedge \mathbf{G}(\mathbf{X}p \leftrightarrow y) \wedge \neg p$$

with the additional output proposition $p$

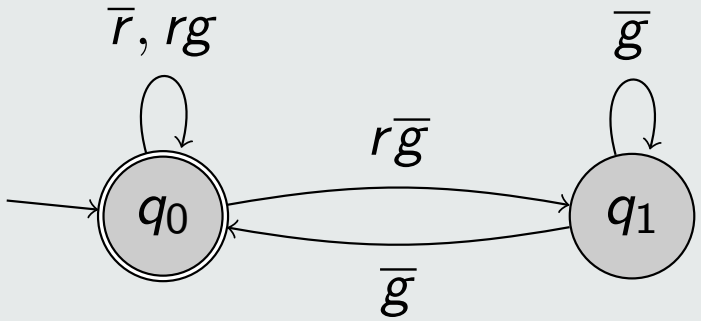
# Encoding deterministic Büchi automata

## Example deterministic Büchi automata for $\mathbf{G}(b \rightarrow \mathbf{F}g)$



# Encoding deterministic Büchi automata

## Example deterministic Büchi automata for $\mathbf{G}(b \rightarrow \mathbf{F}g)$



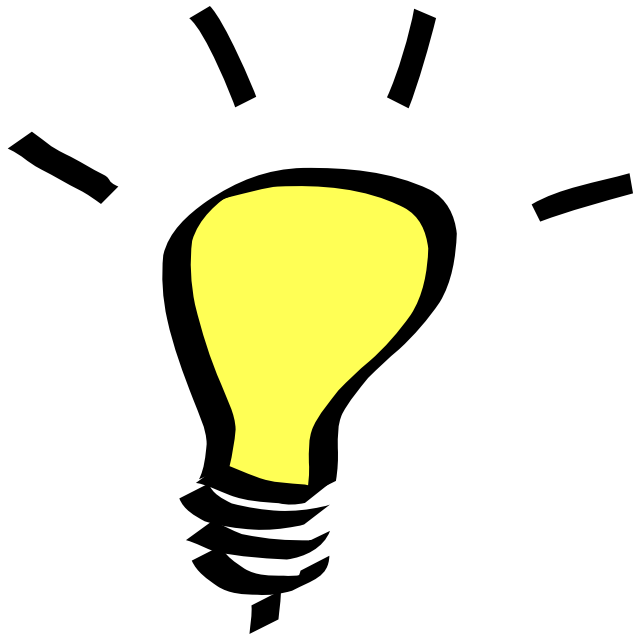

## Slugs specification code (interpreting $\mathbf{G}(b \rightarrow \mathbf{F}g)$ as a guarantee)

```
[OUTPUT]
s

[SYS_INIT]
! s

[SYS_TRANS]
s' <-> ((! g & s) | r & ! g)

[SYS_LIVENESS]
! s'
```

# GR(1) Synthesis – Conclusion

## Summary

- Fast (exponential time) synthesis for simple specification classes
- Approach can be compressed to a single fixed point equation
  $\rightarrow$ allows extensions (e.g., Dathathri et al., 2017; Ehlers, 2011; Wolff et al., 2013, ...)
- Useful for CPS if an environment abstraction is available.



# References I


Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.*, 78(3):911–938, 2012.

Sumanth Dathathri, Scott C. Livingston, and Richard M. Murray. Enhancing tolerance to unexpected jumps in GR(1) games. In *Proceedings of the 8th International Conference on Cyber-Physical Systems, ICCPS 2017, Pittsburgh, Pennsylvania, USA, April 18-20, 2017*, pages 37–47, 2017. doi: 10.1145/3055004.3055014. URL `http://doi.acm.org/10.1145/3055004.3055014`.

Rüdiger Ehlers. Generalized rabin(1) synthesis with applications to robust system synthesis. In *NASA Formal Methods - Third International Symposium, NFM 2011, Pasadena, CA, USA, April 18-20, 2011. Proceedings*, pages 101–115, 2011. doi: 10.1007/978-3-642-20398-5_9. URL `https://doi.org/10.1007/978-3-642-20398-5_9`.

Eric M. Wolff, Ufuk Topcu, and Richard M. Murray. Efficient reactive controller synthesis for a fragment of linear temporal logic. In *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, pages 5033–5040, 2013. doi: 10.1109/ICRA.2013.6631296. URL `https://doi.org/10.1109/ICRA.2013.6631296`.

Tichakorn Wongpiromsarn, Ufuk Topcu, Necmiye Ozay, Huan Xu, and Richard M. Murray. Tulip: a software toolbox for receding horizon temporal logic planning. In *Proceedings of the 14th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2011, Chicago, IL, USA, April 12-14, 2011*, pages 313–314, 2011. doi: 10.1145/1967701.1967747. URL `http://doi.acm.org/10.1145/1967701.1967747`.