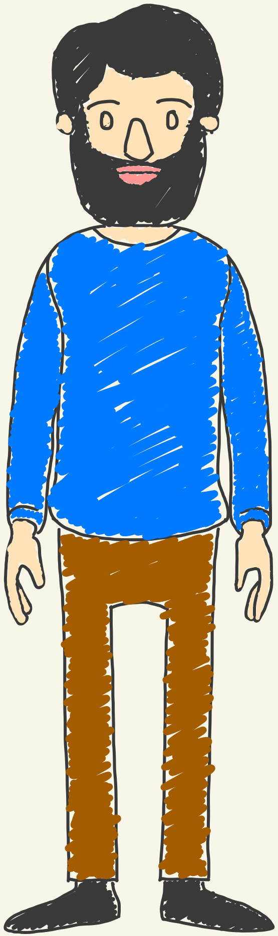


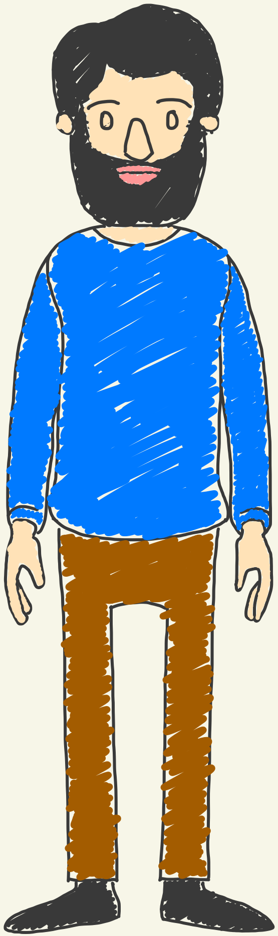
To Assume, Or Not To Assume

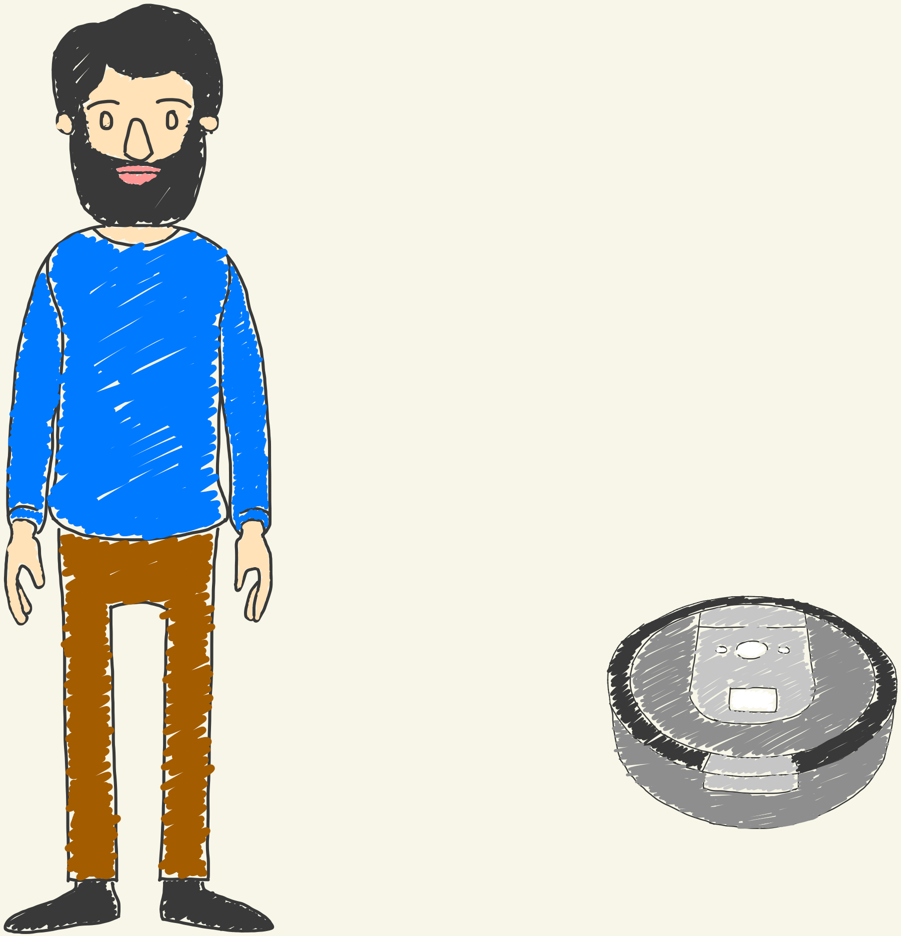
[Computing Adequately Permissive Assumptions for Synthesis]

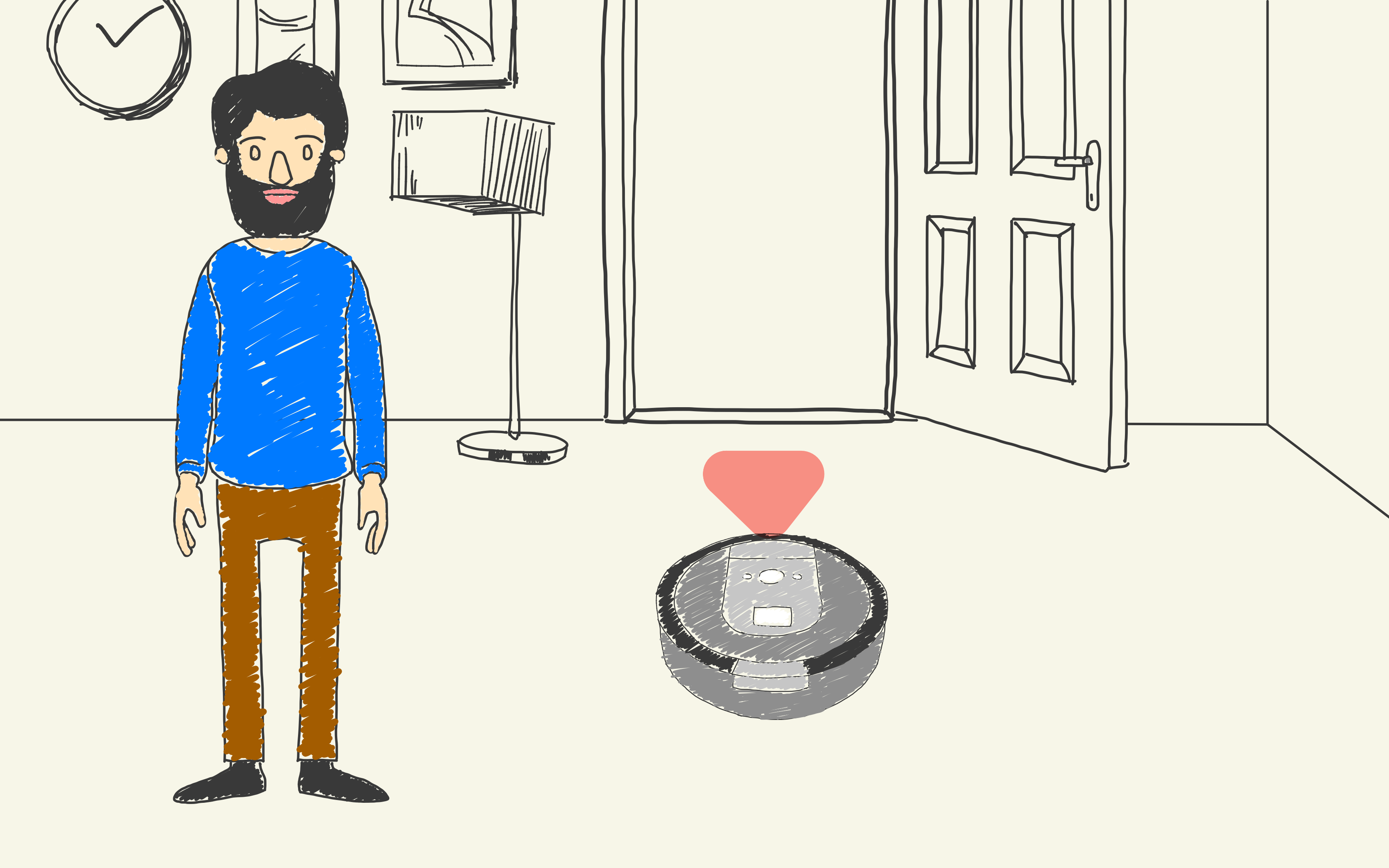
TACAS '23

Ashwani Anand
MPI-SWS

with K. Mallik (ISTA), S.P. Nayak (MPI-SWS), and A.-K. Schmuck (MPI-SWS)





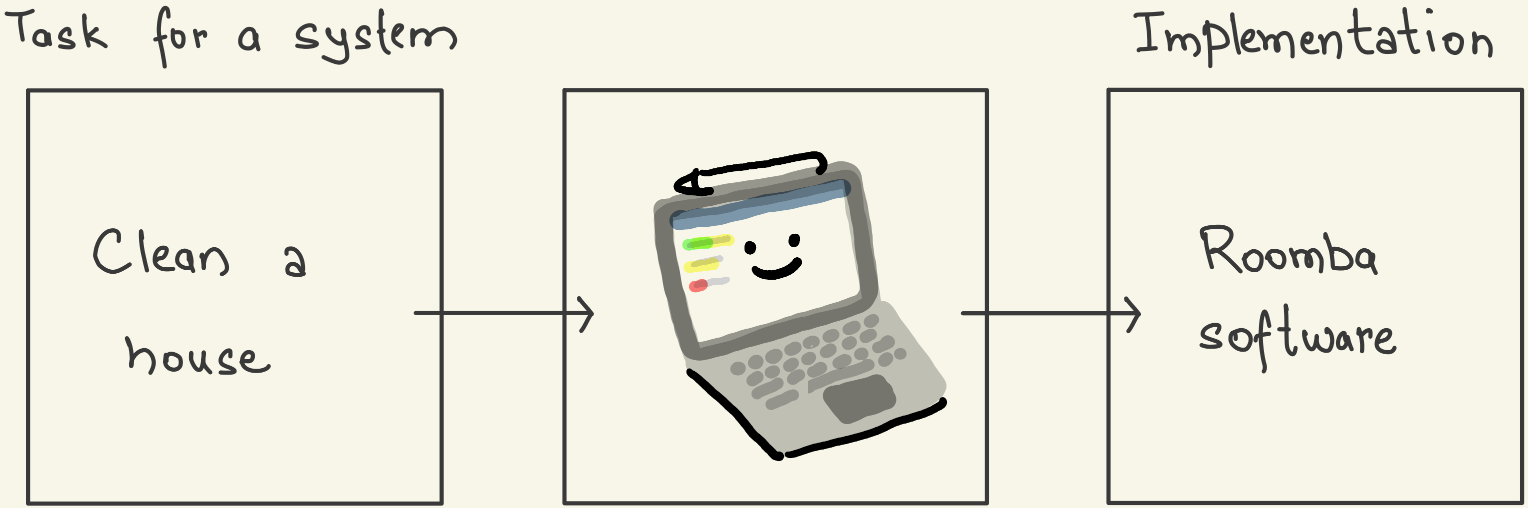



No cleaning strategy
may exist if Environment
is adversarial.


No cleaning strategy may exist if Environment is adversarial.
Environment rarely acts adversarially.


No cleaning strategy may exist if Environment is adversarial.
Humans may know this assumption.
Computers don't.
Environment rarely acts adversarially.

Reactive Synthesis



Reactive Synthesis

Task for a system

Clean a house

Implementation

Roomba software

✔ Works well if no assumption is needed.

Reactive Synthesis

Task for a system | | Implementation

Clean a house →  → Roomba software

Works well if no assumption is needed.

Might fail without some assumptions.
E.g. the user will not block the path.

What has been done?

✓ Chatterjee et. al. [CONCUR'08] introduce the notion of assumption for games on graphs.

What has been done?

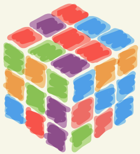
Chatterjee et. al. [CONCUR'08] introduce the notion of assumption for games on graphs.

Their method requires solving NP-hard problem.

What has been done?

Chatterjee et. al. [CONCUR'08] introduce the notion of assumption for games on graphs.

Their method requires solving NP-hard problem.

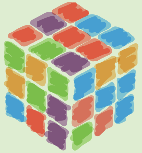


Fails to give a sufficient assumption, even if it exists.

What has been done?

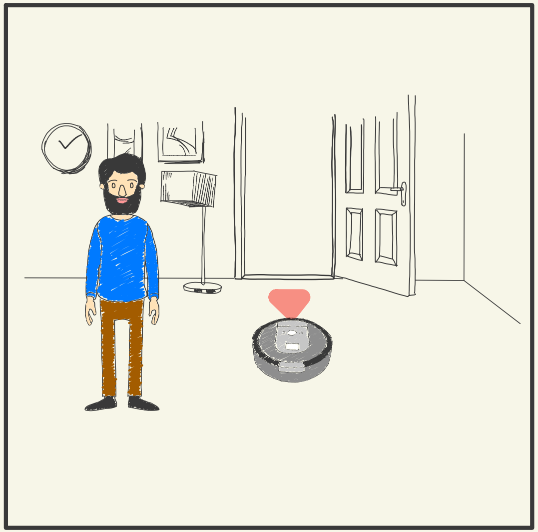
Chatterjee et. al. [CONCUR'08] introduce the notion of assumption for games on graphs.

Their method requires solving NP-hard problem.

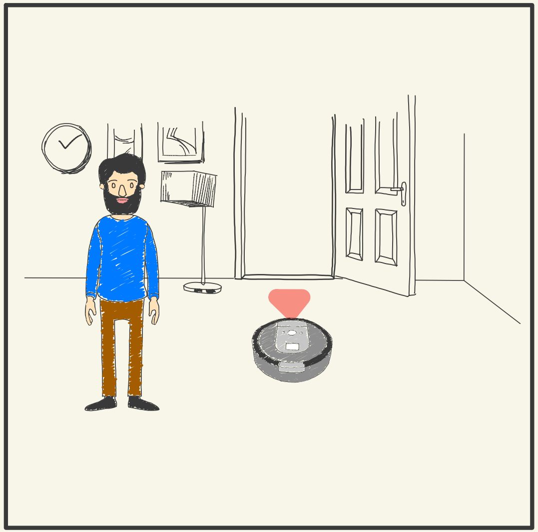


Fails to give a sufficient assumption, even if it exists.

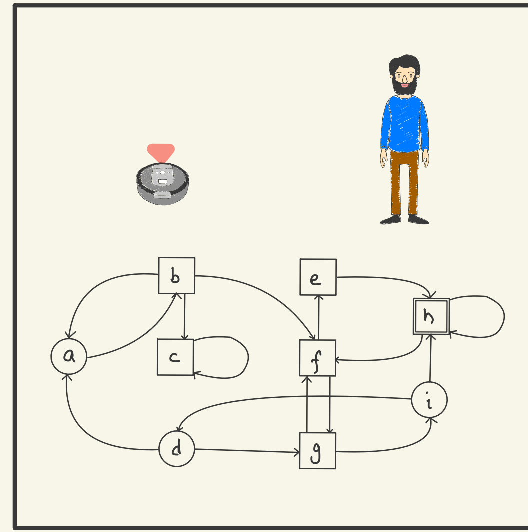
Reactive Synthesis : The Standard



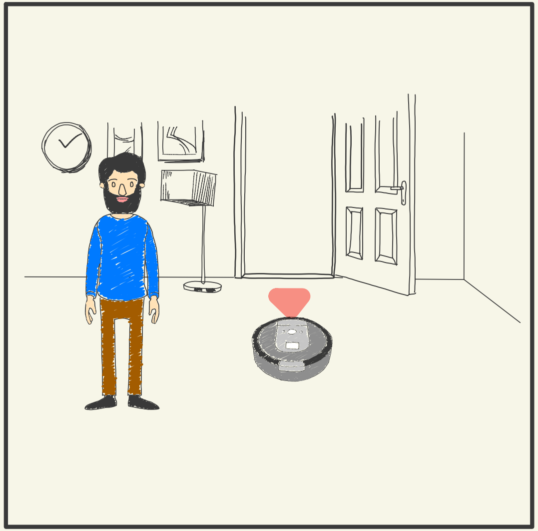
Reactive Synthesis : The Standard



Convert to a game on graph

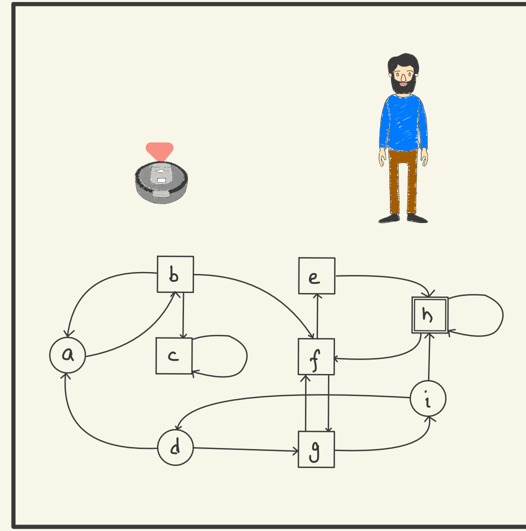


Reactive Synthesis : The Standard

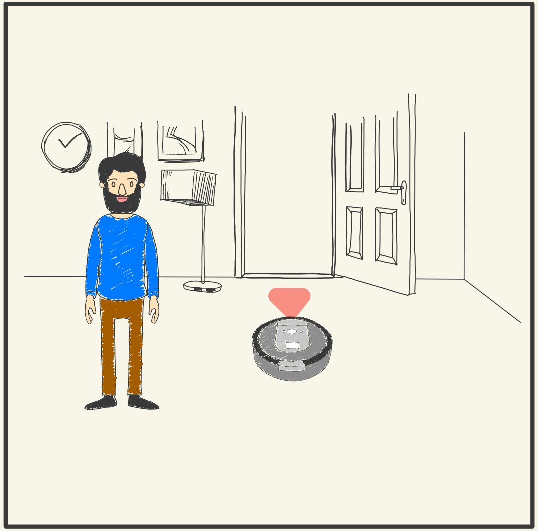


Convert to a game on graph

Winning strategy acts as the software.

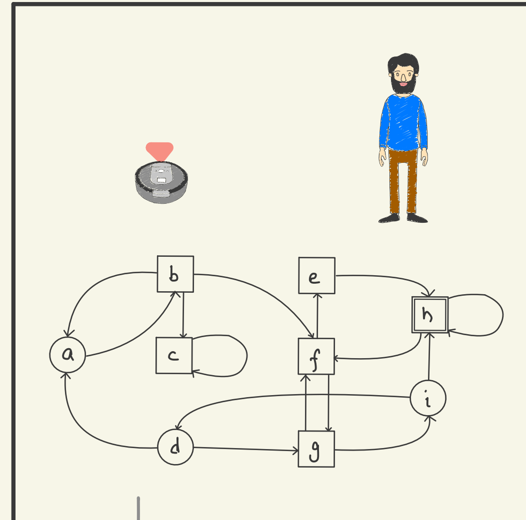


Reactive Synthesis : The Standard



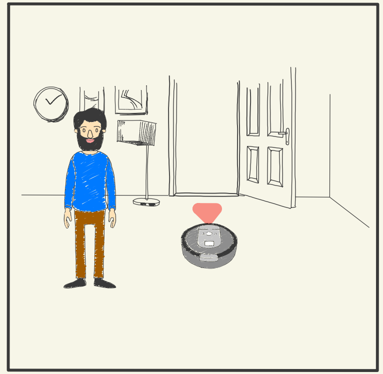
Convert to a game on graph

Winning strategy acts as the software.

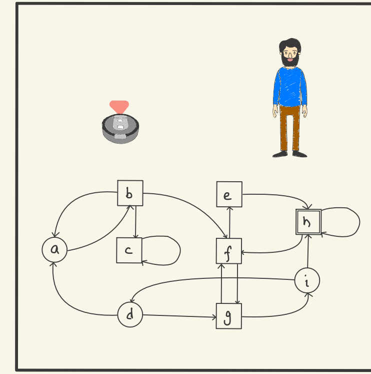


find assumptions on the environment via the game graph.

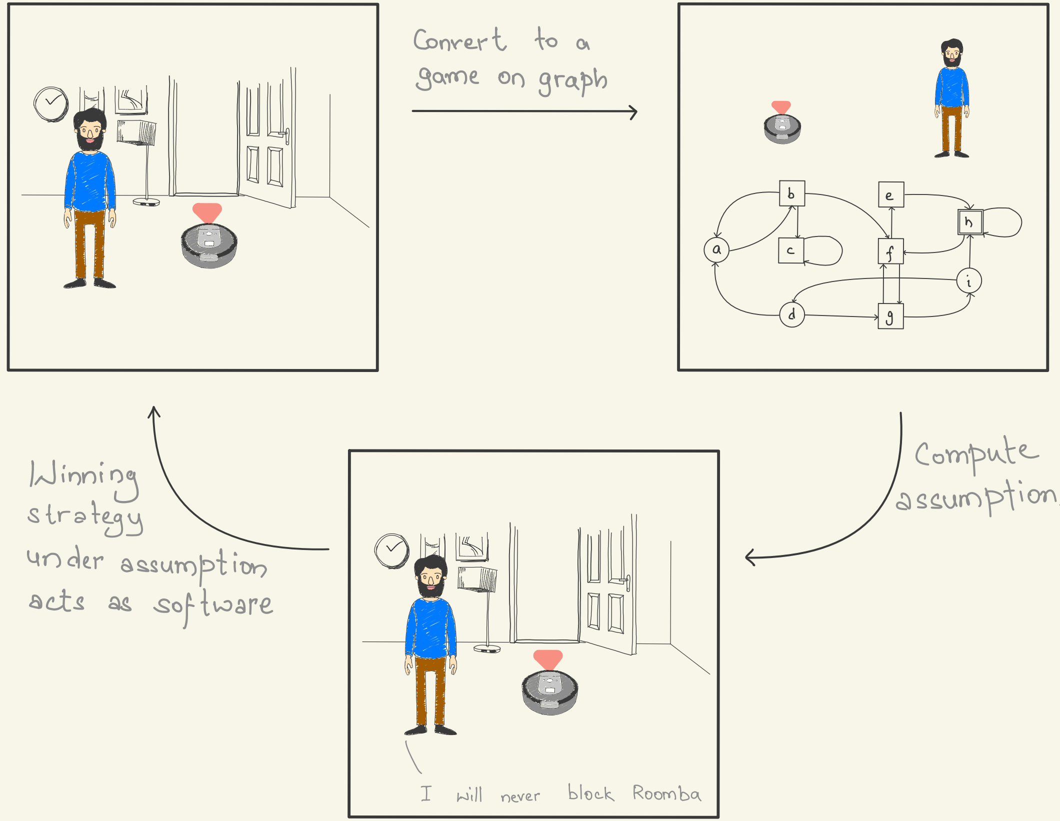
The Return of Reactive Synthesis



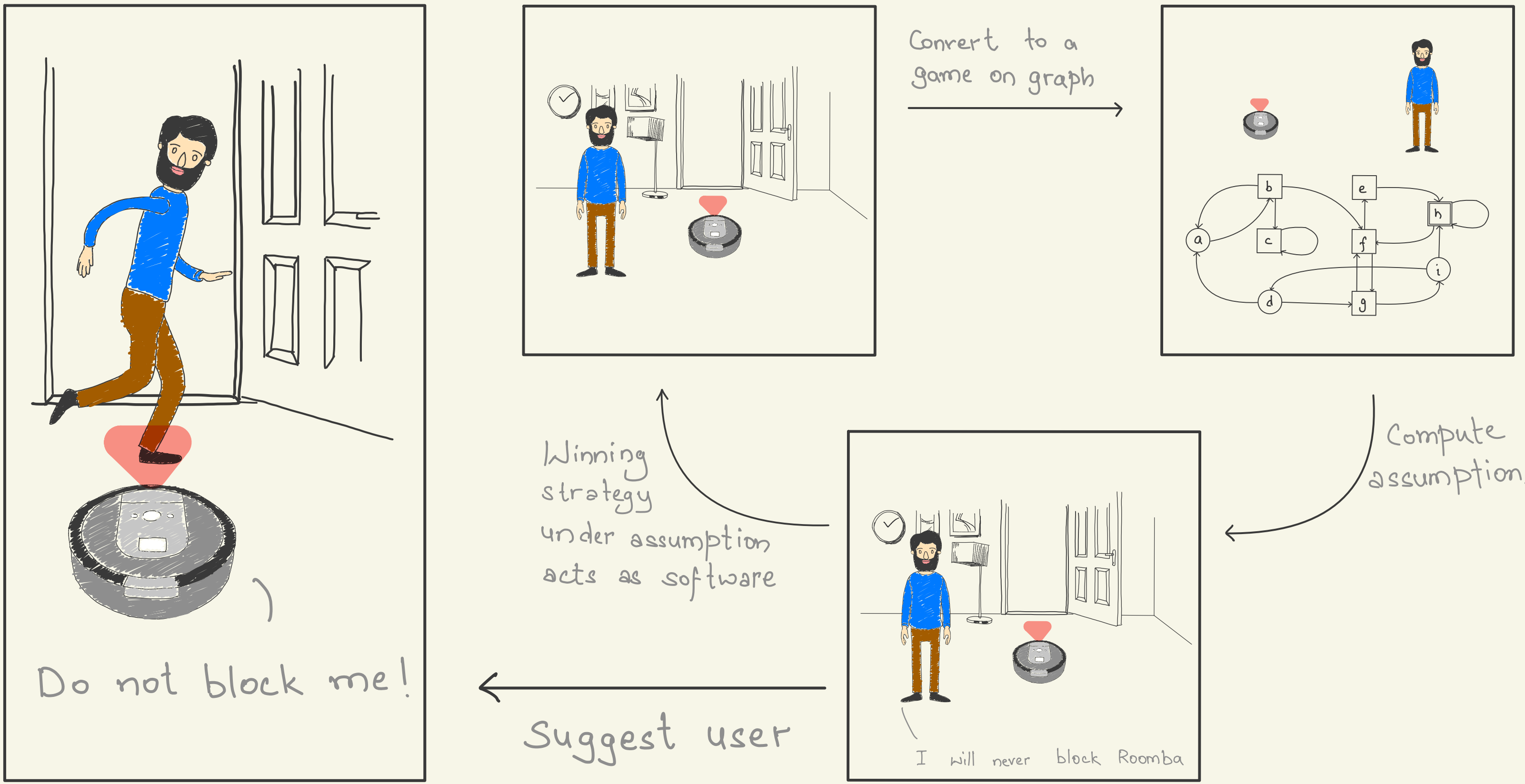
Convert to a game on graph



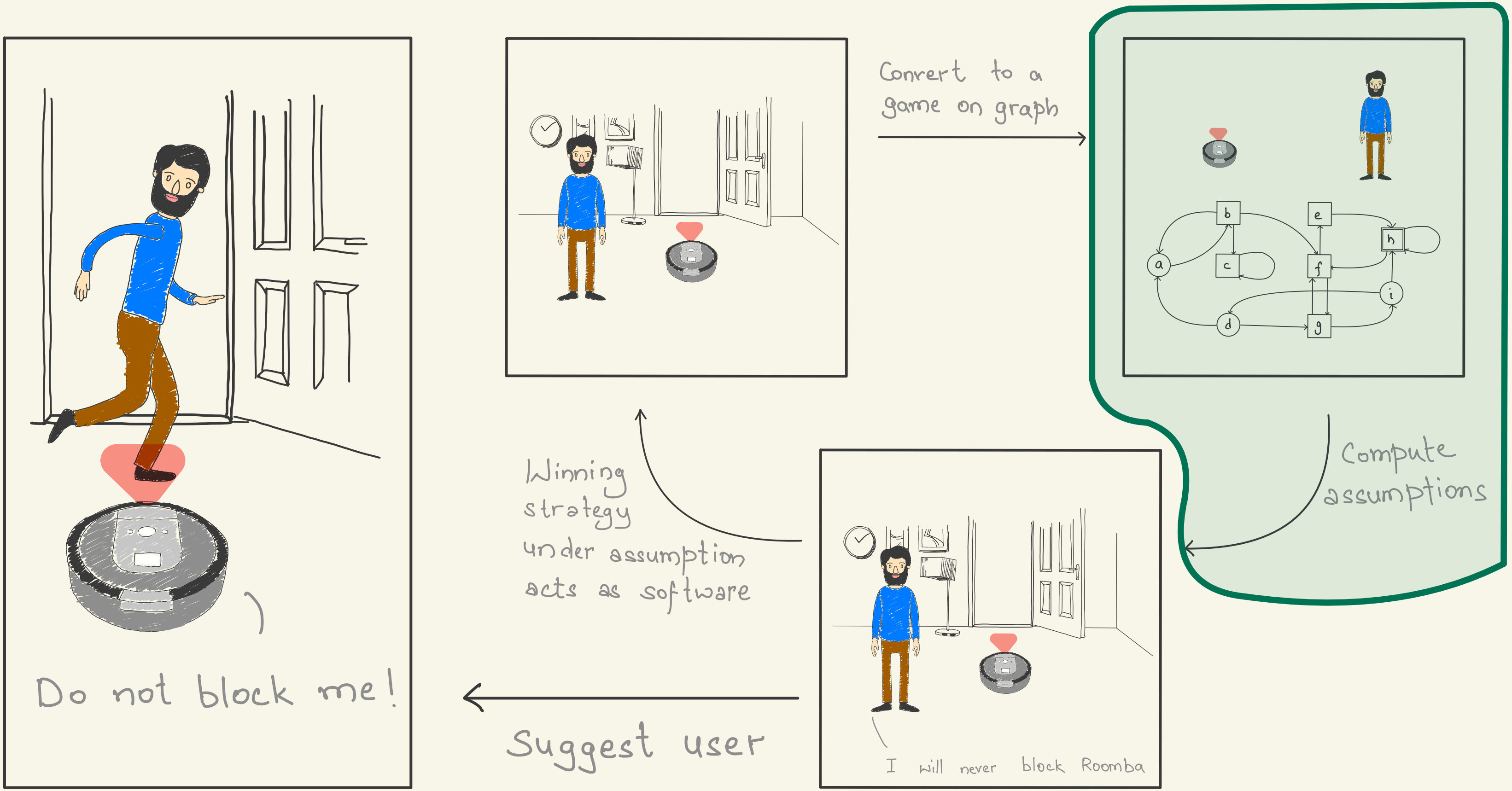
The Return of Reactive Synthesis



The Return of Reactive Synthesis



The Return of Reactive Synthesis



Precap

Assumptions | computation

Precap

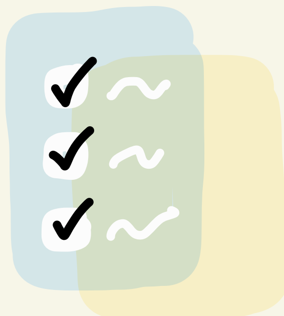
Assumptions computation

Permissive

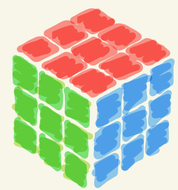
Precap

Assumptions computation

Novel Templates



Permissive

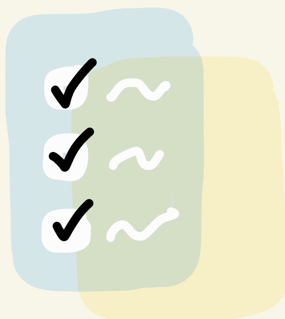


Complete

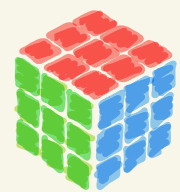
Precap

Assumptions computation

Novel Templates



Permissive



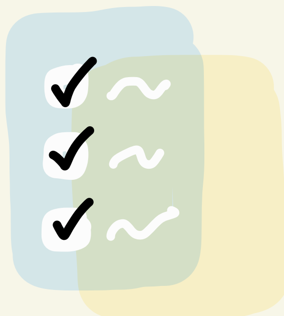
Complete

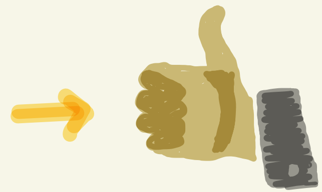
Faster

Precap

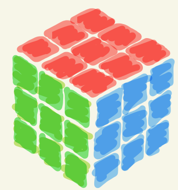
Assumptions computation

Novel Templates





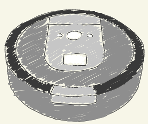
Permissive

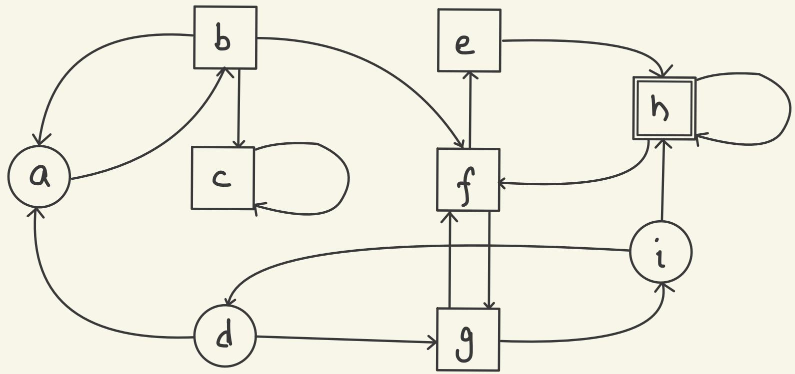


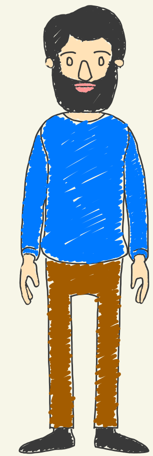
Complete

Faster

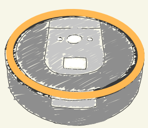
Games on Graphs

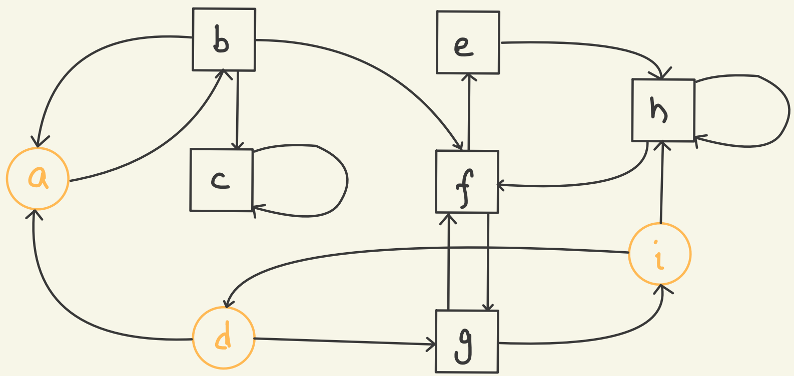


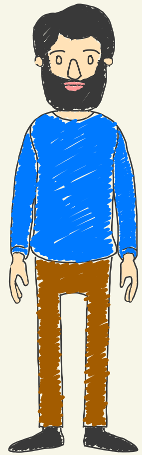




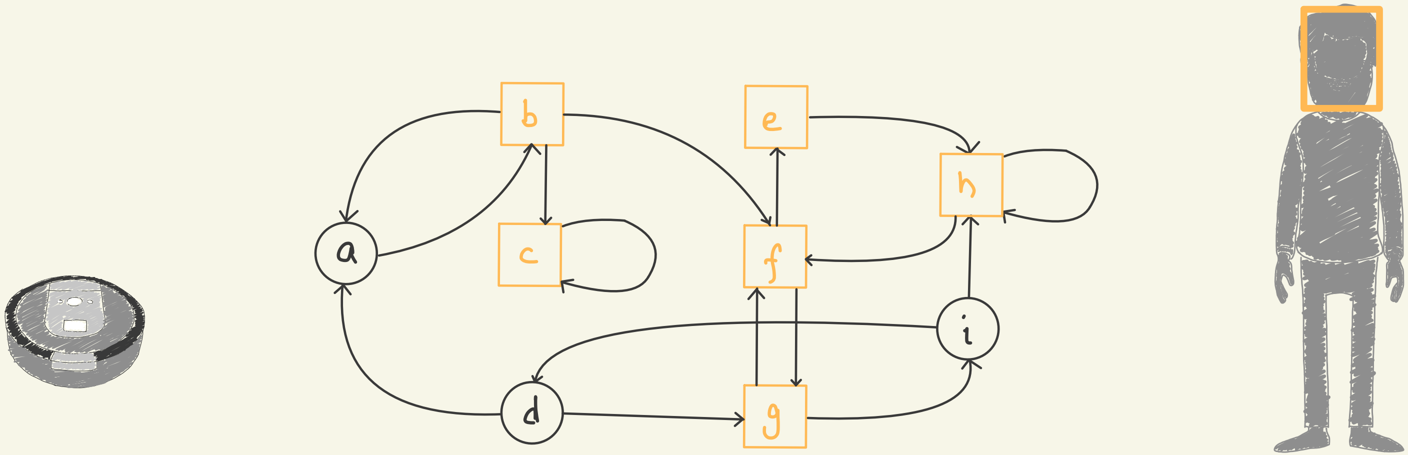
Games on Graphs





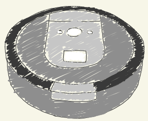


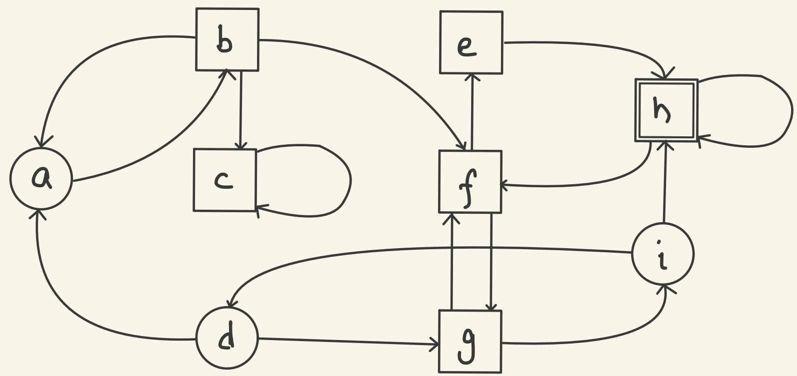
Games on Graphs

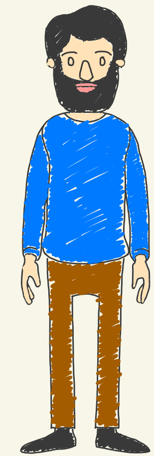


Games on Graphs

Always eventually visit h

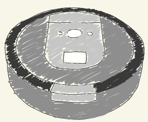


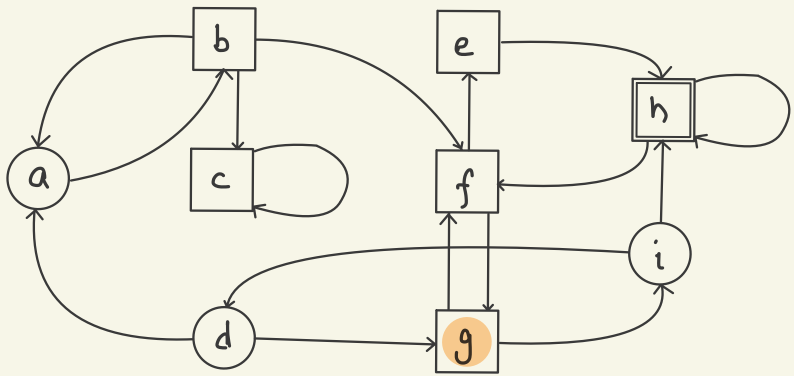


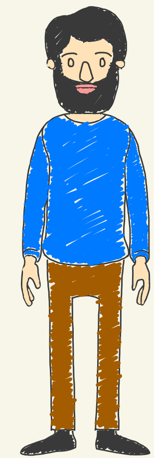


Games on Graphs

Always eventually visit h



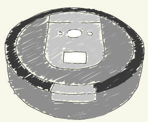


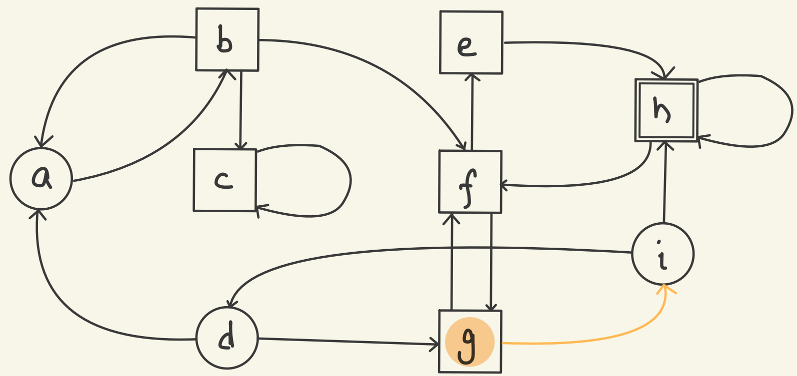


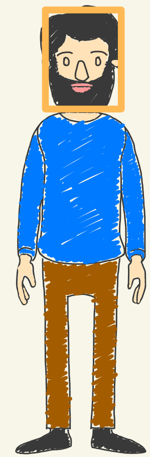
g

Games on Graphs

Always eventually visit h



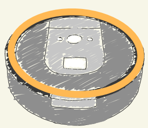


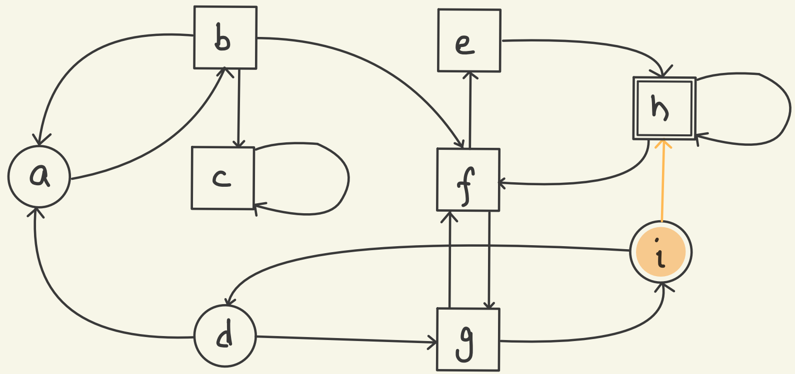


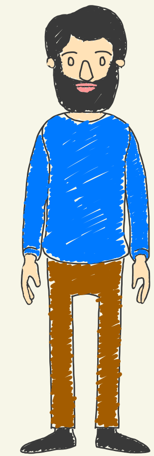
$g \to i$

Games on Graphs

Always eventually visit h



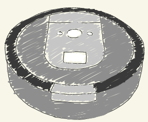


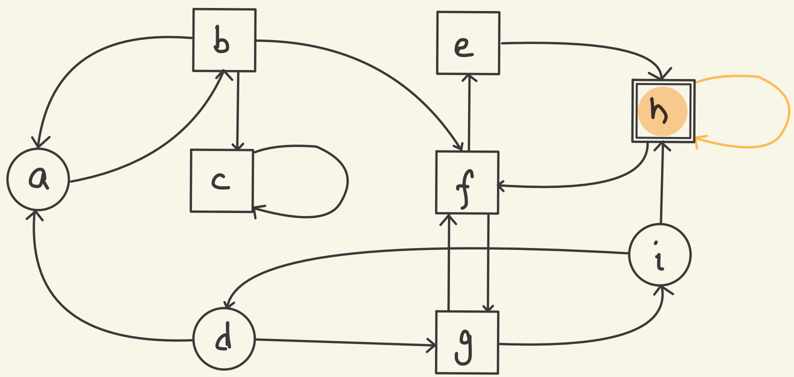


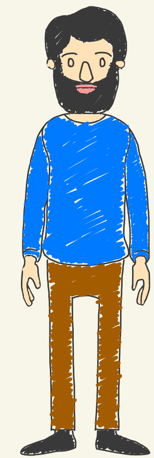
$g \to i \to h$

Games on Graphs

Always eventually visit h



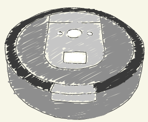


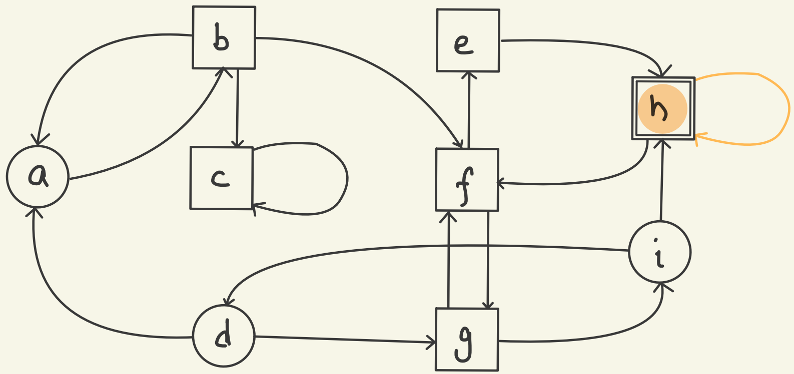


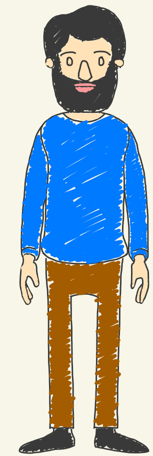
$g \to i \to h \to h$

Games on Graphs

Always eventually visit h



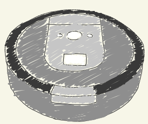


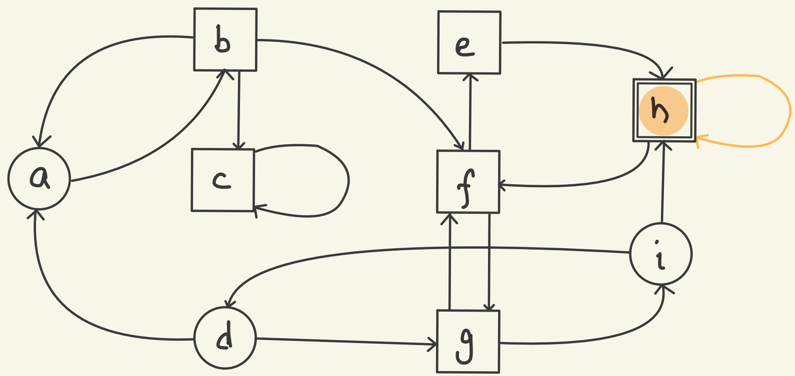


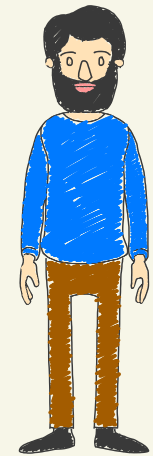
$g \to i \to h \to h \to h$

Games on Graphs

Always eventually visit h



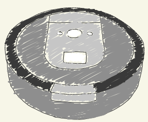


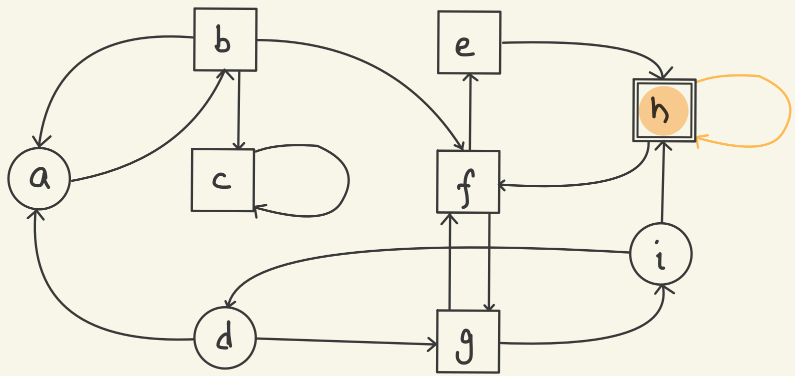


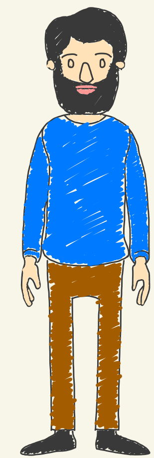
$$g \to i \to h \to h \to h \to \cdots$$

Games on Graphs

Always eventually visit h



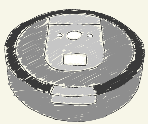


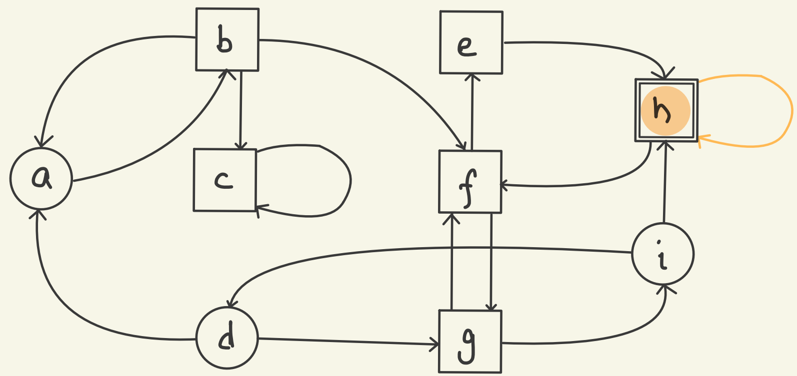


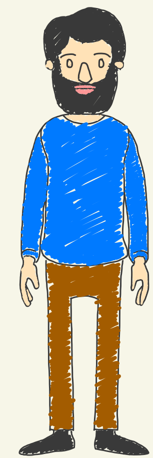
$g \to i \to h \to h \to h \to \cdots$ ✓

Games on Graphs

Always eventually visit h







$g \to i \to h \to h \to h \to \cdots$

Assumptions restrict the choices of Environment

Assumptions on Environment

LTL formula ψ on vertices of the game graph

Assumptions on Environment

LTL formula ψ on vertices of the game graph

Sufficient

If environment satisfies assumption,
system can finish the task

Assumptions on Environment

LTL formula ψ on vertices of the game graph

Sufficient

If environment satisfies assumption, system can finish the task

Assumptions on Environment

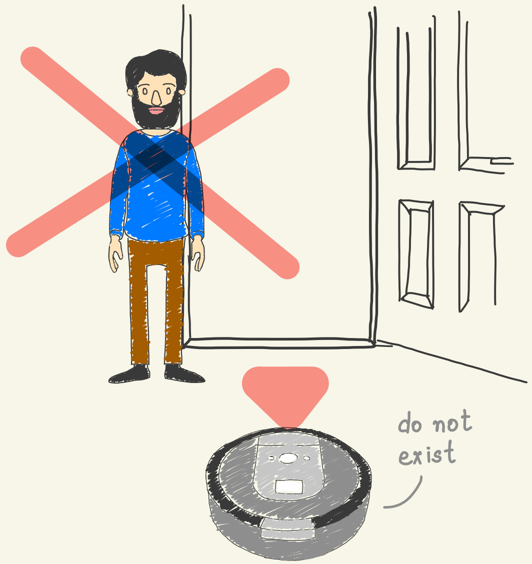
LTL formula ψ on vertices of the game graph

Sufficient

If environment satisfies assumption, system can finish the task

Implementable

Environment can satisfy the assumption



Assumptions on Environment

LTL formula φ on vertices of the game graph

Sufficient

If environment satisfies assumption,
system can finish the task

Implementable

Environment can satisfy the assumption

Assumptions on Environment

LTL formula ψ on vertices of the game graph

Sufficient

If environment satisfies assumption, system can finish the task

Implementable

Environment can satisfy the assumption

Permissive

Assumption does not restrict the environment too much

Assumptions on Environment

LTL formula ψ on vertices of the game graph

Sufficient

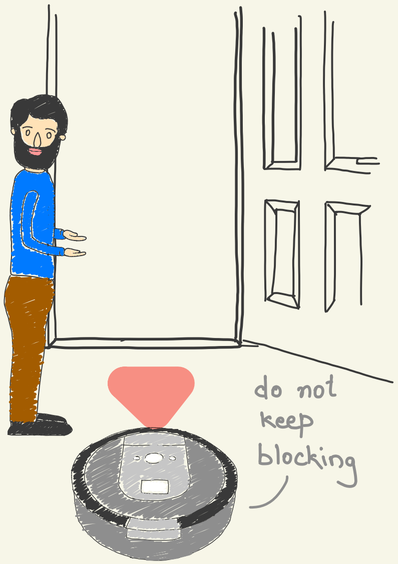
If environment satisfies assumption, system can finish the task

Implementable

Environment can satisfy the assumption

Permissive

Assumption does not restrict the environment too much



Assumptions on Environment

LTL formula ψ on vertices of the game graph

Sufficient

If environment satisfies assumption, system can finish the task

Implementable

Environment can satisfy the assumption

Permissive

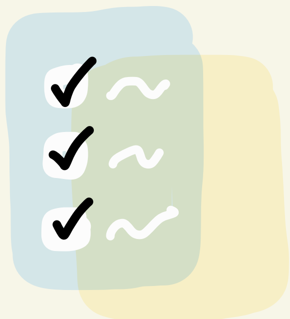
Assumption does not restrict the environment too much

Adequately Permissive Assumption

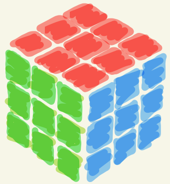
Checkpoint

Assumptions computation

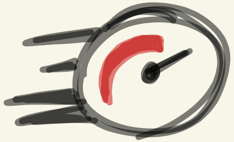
Novel Templates



Permissive

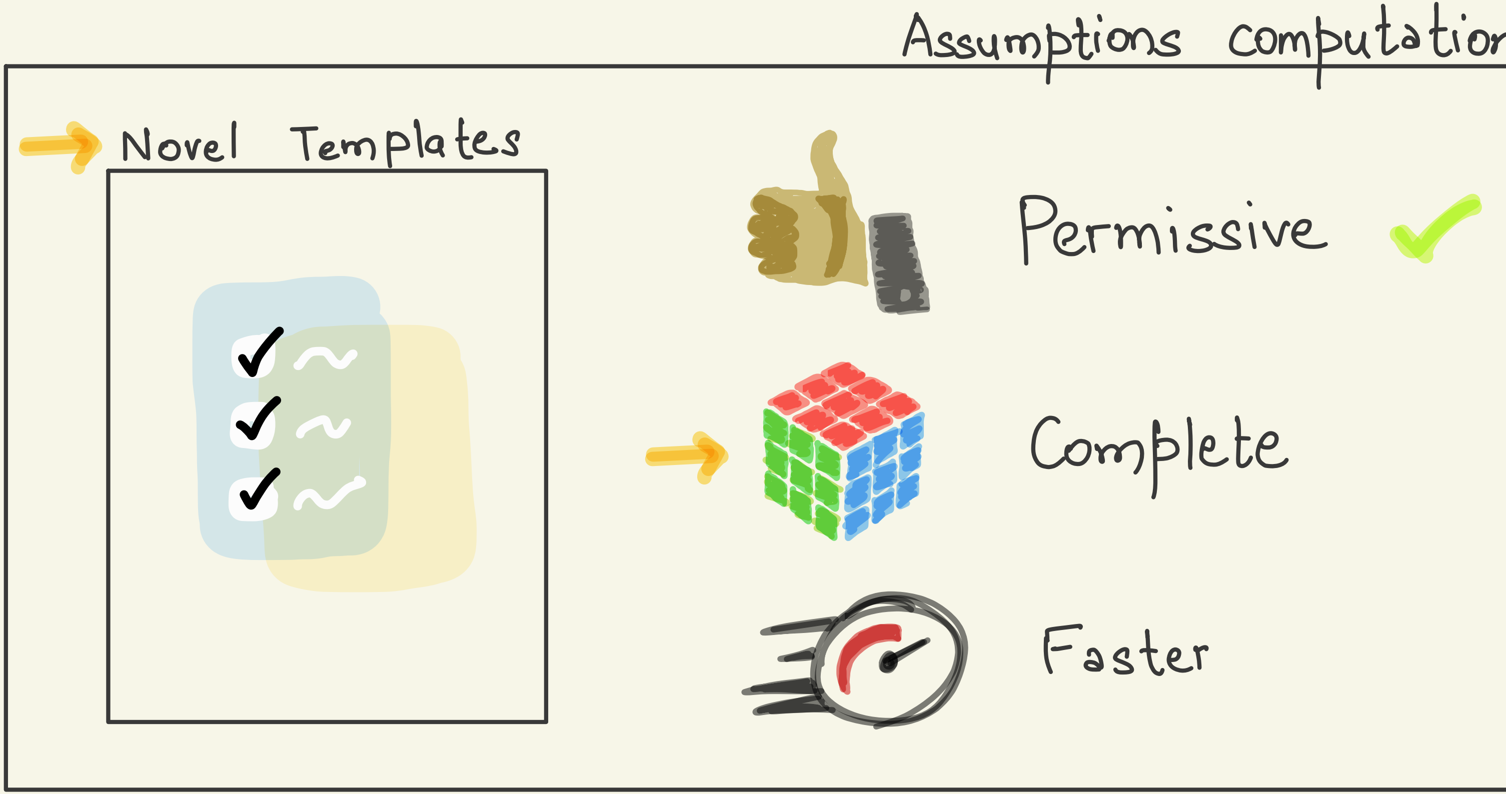


Complete



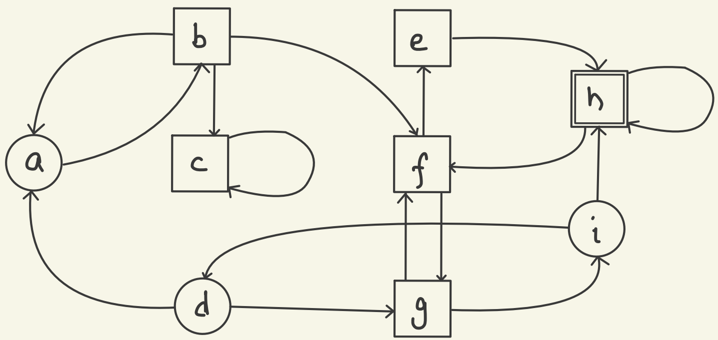
Faster

Checkpoint


Assumptions computation
Novel Templates
Permissive
Complete
Faster

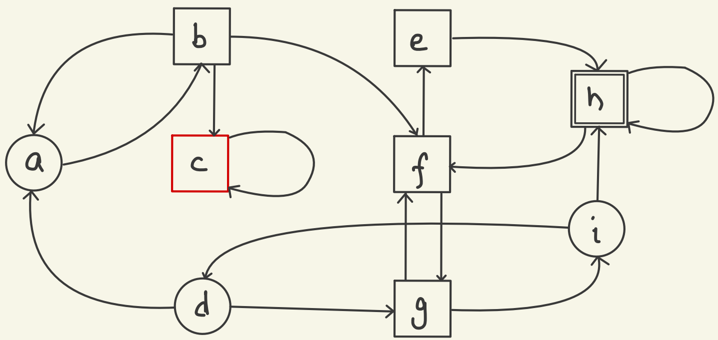
Computing Adequately Permissive Assumptions

Büchi objective: Always eventually visit $\boxed{}$



Computing Adequately Permissive Assumptions

Büchi objective: Always eventually visit

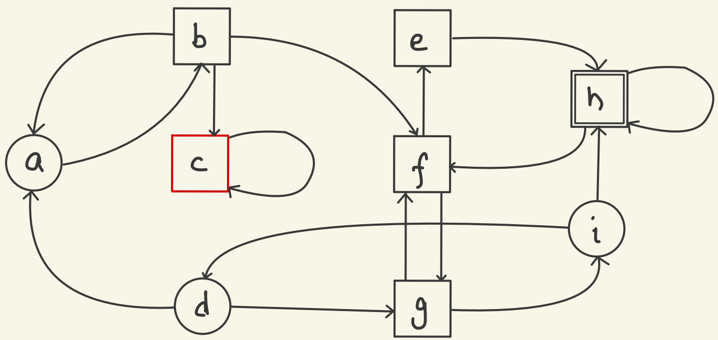


No way of satisfying the objective from $\boxed{c}$.

Hence, it must never be visited.

Computing Adequately Permissive Assumptions

Büchi objective: Always eventually visit $\boxed{}$

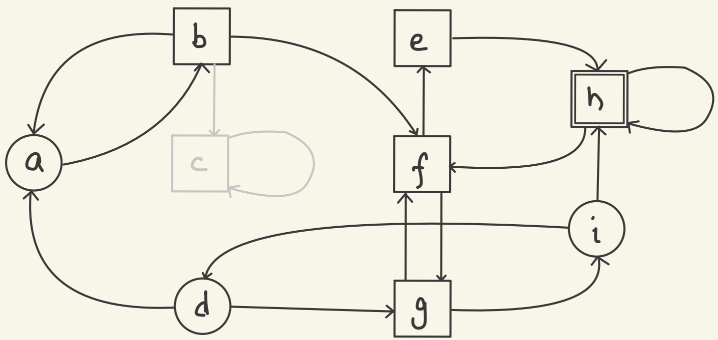


No way of satisfying the objective from $\boxed{c}$. Hence, it must never be visited.

- Always not $b \to c$.

Computing Adequately Permissive Assumptions

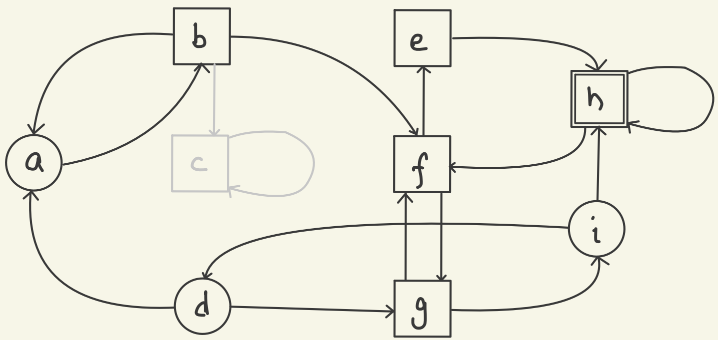
Büchi objective: Always eventually visit $\boxed{\square}$



- Always not $b \to c$.

Computing Adequately Permissive Assumptions

Büchi objective: Always eventually visit $\boxed{\boxed{}}$

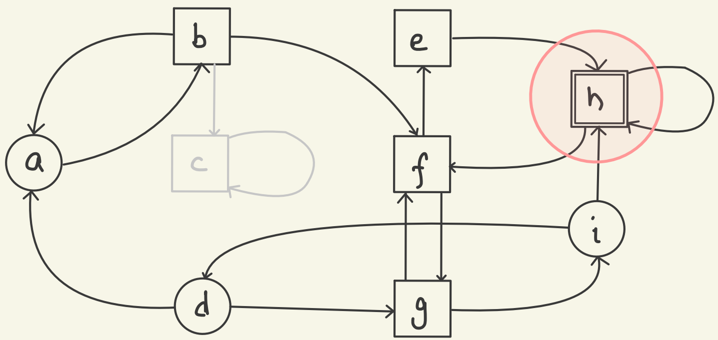


Enough to compute assumptions to "reach" h from remaining vertices

- Always not $b \to c$.

Computing Adequately Permissive Assumptions

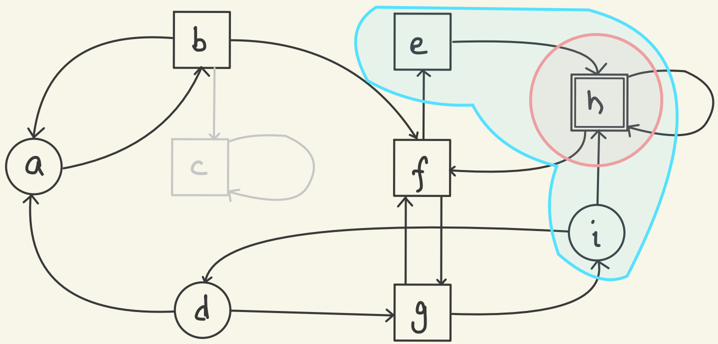
Büchi objective: Always eventually visit $\boxed{\square}$



- Always not $b \to c$.

Computing Adequately Permissive Assumptions

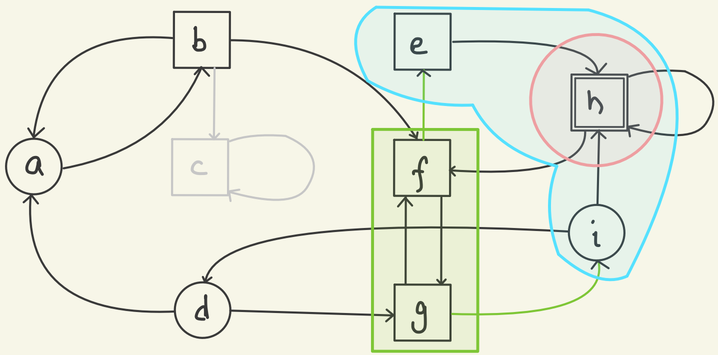
Büchi objective: Always eventually visit $\boxed{}$



- Always not b→c.

Computing Adequately Permissive Assumptions

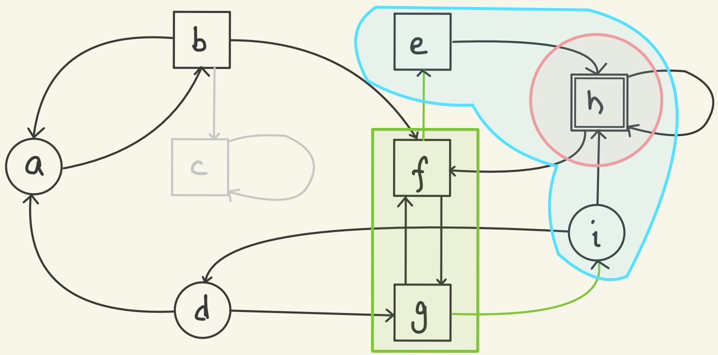
Büchi objective: Always eventually visit □



- Always not $b \to c$.

Computing Adequately Permissive Assumptions

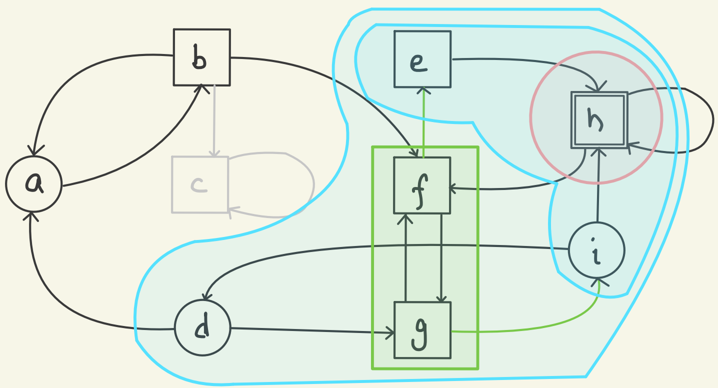
Büchi objective: Always eventually visit □



- Always not $b \to c$.
- Always eventually $\{f, g\} \Rightarrow$ always eventually $f \to e$ or $g \to i$

Computing Adequately Permissive Assumptions

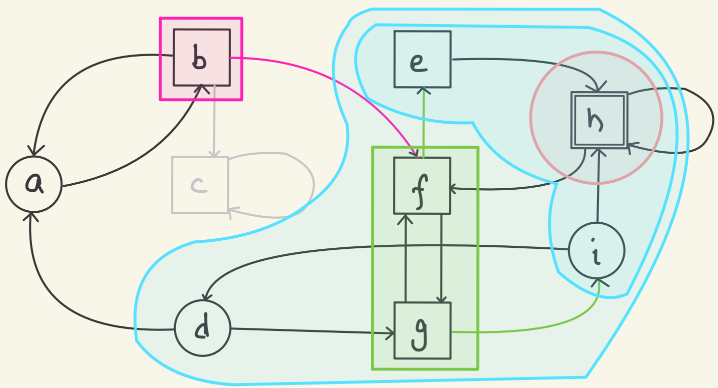
Büchi objective: Always eventually visit $\boxed{\boxed{}}$



- Always not $b \to c$.
- Always eventually $\{f, g\} \Rightarrow$ always eventually $f \to e$ or $g \to i$

Computing Adequately Permissive Assumptions

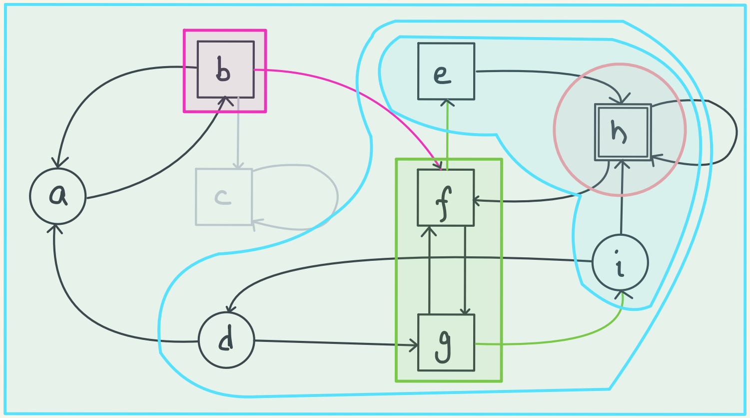
Büchi objective: Always eventually visit $\boxed{}$



- Always not b → c.
- Always eventually {f, g} ⇒ always eventually f → e or g → i
- Always eventually { b } ⇒ always eventually b → f

Computing Adequately Permissive Assumptions

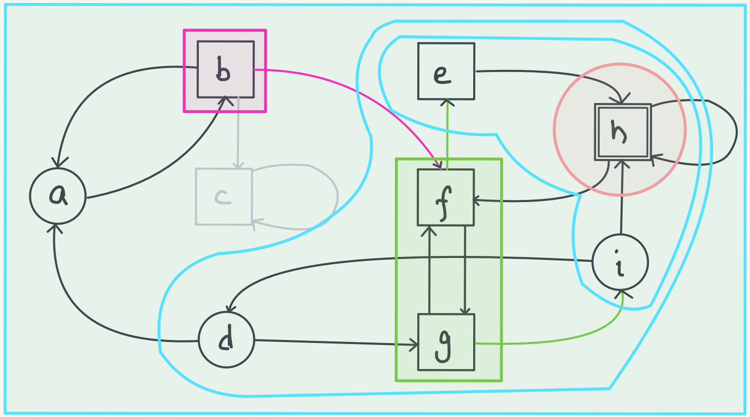
Büchi objective: Always eventually visit $\boxed{}$



- Always not $b \to c$.
- Always eventually $\{f, g\} \Rightarrow$ always eventually $f \to e$ or $g \to i$
- Always eventually $\{ b \} \Rightarrow$ always eventually $b \to f$

Computing Adequately Permissive Assumptions

Büchi objective: Always eventually visit $\boxed{\square}$



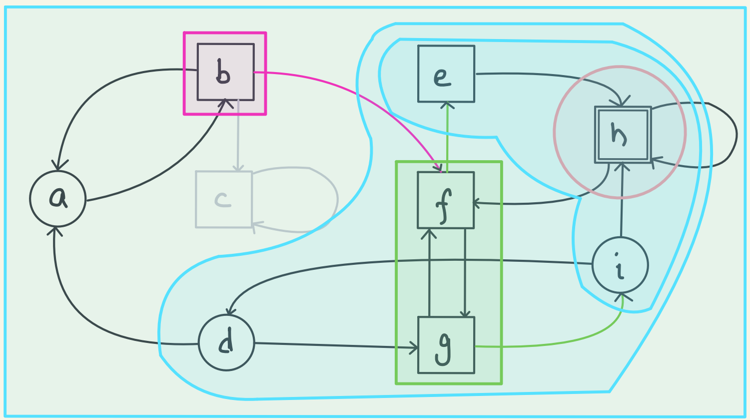
Safety template

- Always not $b \to c$.
- Always eventually $\{f, g\} \Rightarrow$ always eventually $f \to e$ or $g \to i$
- Always eventually $\{ b \} \Rightarrow$ always eventually $b \to f$

Group liveness templates

Computing Adequately Permissive Assumptions

Büchi objective: Always eventually visit $\square$



Runs in time $O(m+n)$

edges

vertices

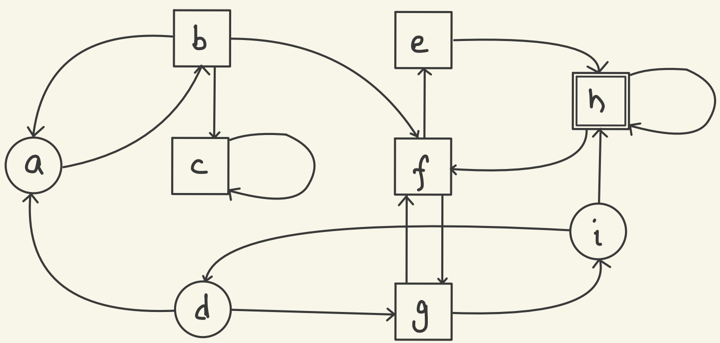
Safety template

- Always not $b \to c$.
- Always eventually $\{f, g\} \Rightarrow$ always eventually $f \to e$ or $g \to i$
- Always eventually $\{\ b\ \} \Rightarrow$ always eventually $b \to f$

Group liveness templates

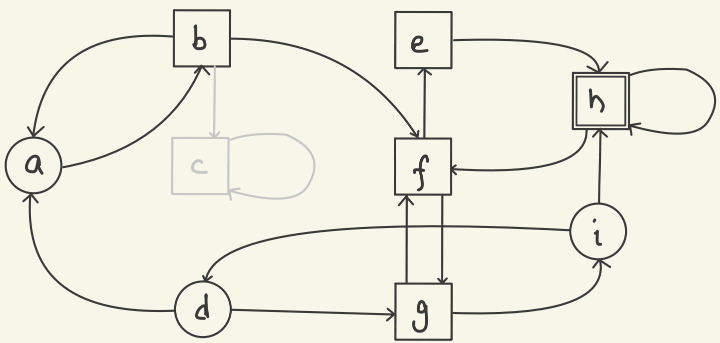
Computing Adequately Permissive Assumptions

coBüchi objective : Eventually always visit



Computing Adequately Permissive Assumptions

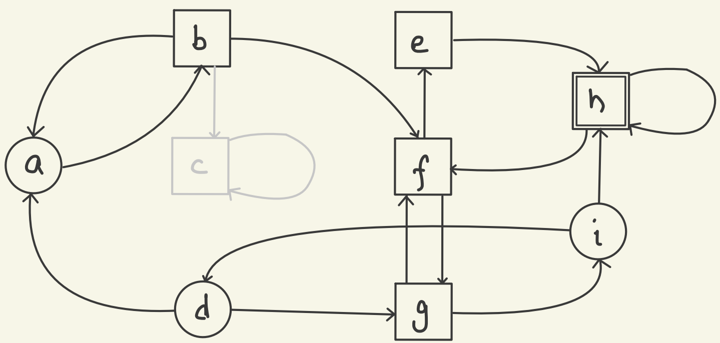
coBüchi objective : Eventually always visit $\boxed{}$



- Always not $b \to c$.

Computing Adequately Permissive Assumptions

coBüchi objective : Eventually always visit □

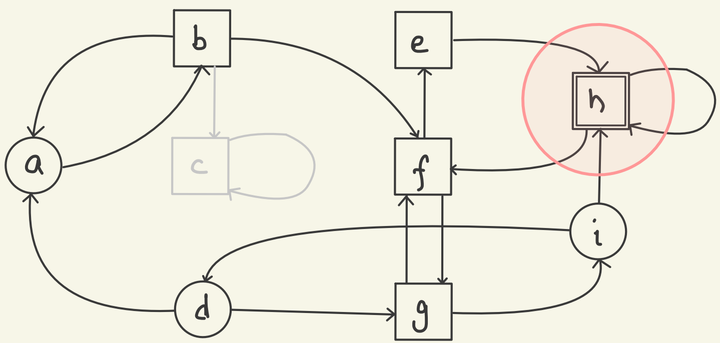


Need to restrict from going further away from $\boxed{h}$ eventually.

- Always not $b \to c$.

Computing Adequately Permissive Assumptions

coBüchi objective : Eventually always visit $\boxed{\boxed{}}$

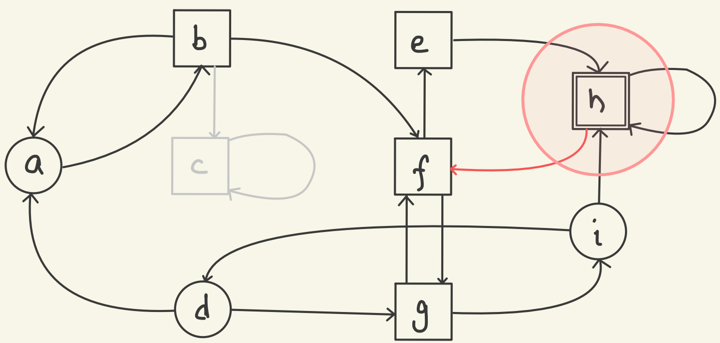


Need to restrict from going further away from $\boxed{\boxed{h}}$ eventually.

- Always not b→c.

Computing Adequately Permissive Assumptions

coBüchi objective : Eventually always visit $\boxed{\boxed{}}$

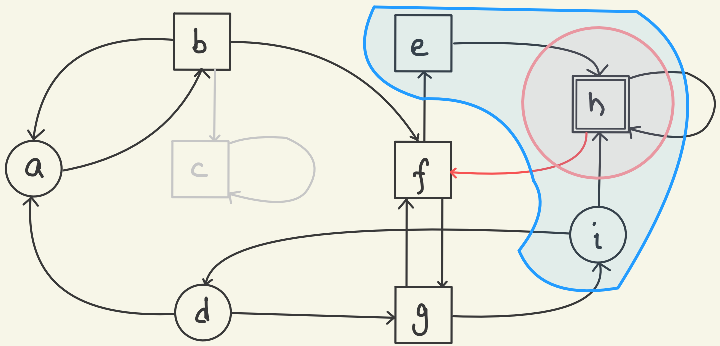


Need to restrict from going further away from $\boxed{\boxed{h}}$ eventually.

- Always not $b \rightarrow c$.
- Eventually always not $h \rightarrow f$

Computing Adequately Permissive Assumptions

coBüchi objective : Eventually always visit $\boxed{}$

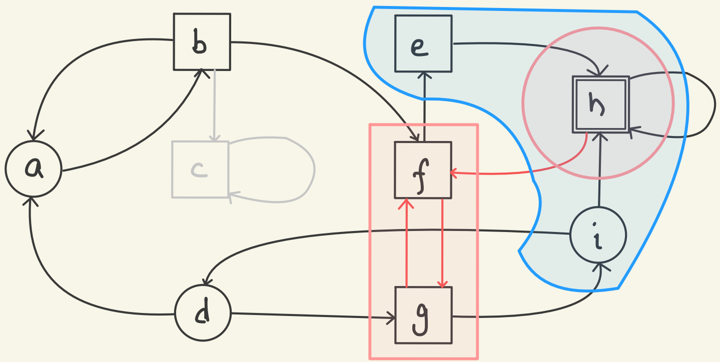


Need to restrict from going further away from $\boxed{h}$ eventually.

- Always not b → c.
- Eventually always not h → f

Computing Adequately Permissive Assumptions

coBüchi objective : Eventually always visit $\boxed{}$

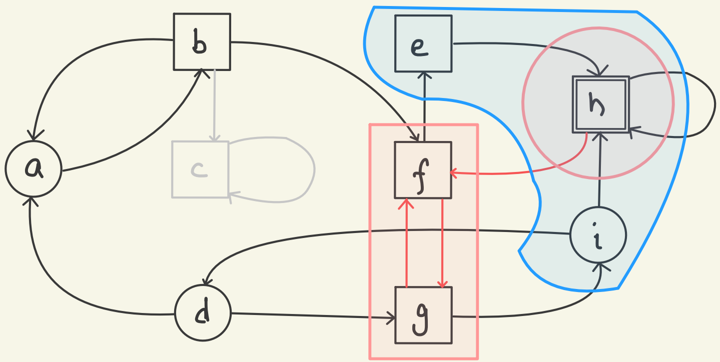


Need to restrict from going further away from $\boxed{h}$ eventually.

- Always not $b \rightarrow c$.
- Eventually always not $h \rightarrow f$

Computing Adequately Permissive Assumptions

coBüchi objective : Eventually always visit □

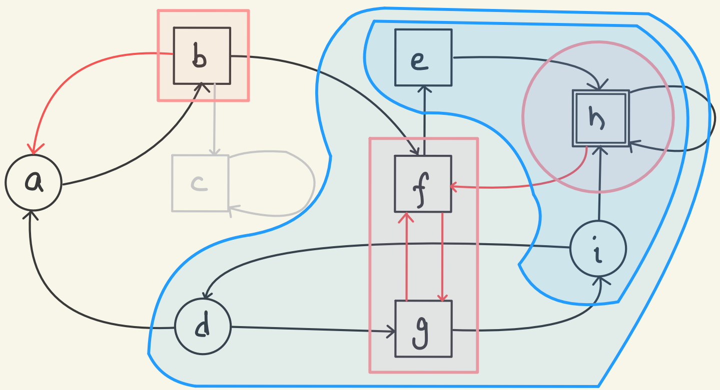


Need to restrict from going further away from h eventually.

- Always not b → c.
- Eventually always not h → f
- Eventually always not f → g
- Eventually always not g → f

Computing Adequately Permissive Assumptions

coBüchi objective : Eventually always visit $\boxed{}$

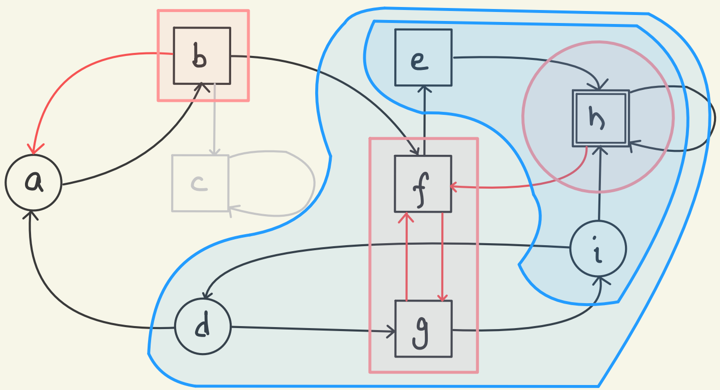


Need to restrict from going further away from $\boxed{h}$ eventually.

- Always not b → c.
- Eventually always not h → f
- Eventually always not f → g
- Eventually always not g → f

Computing Adequately Permissive Assumptions

coBüchi objective : Eventually always visit □

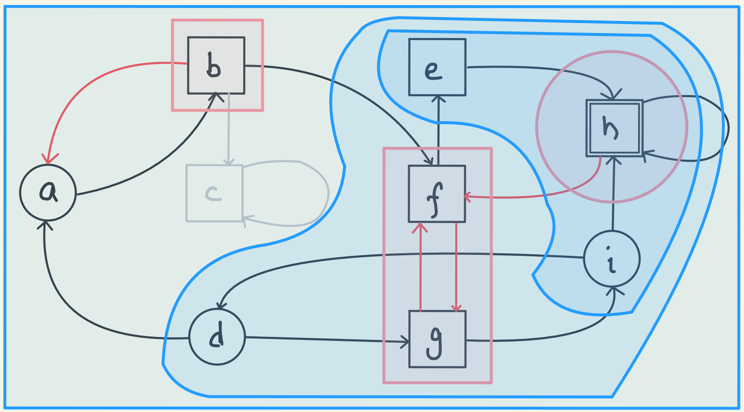


Need to restrict from going further away from $\boxed{h}$ eventually.

- Always not $b \to c$.
- Eventually always not $h \to f$
- Eventually always not $f \to g$
- Eventually always not $g \to f$
- Eventually always not $b \to a$

Computing Adequately Permissive Assumptions

coBüchi objective : Eventually always visit $\square$

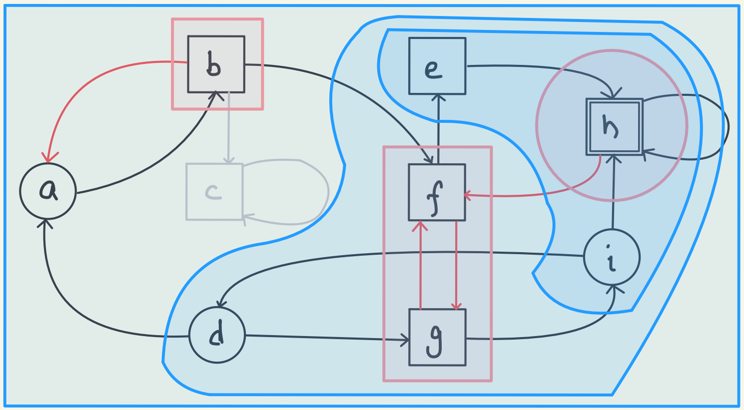


Need to restrict from going further away from $\boxed{h}$ eventually.

- Always not $b \to c$.
- Eventually always not $h \to f$
- Eventually always not $f \to g$
- Eventually always not $g \to f$
- Eventually always not $b \to a$

Computing Adequately Permissive Assumptions

coBüchi objective : Eventually always visit $\boxed{}$



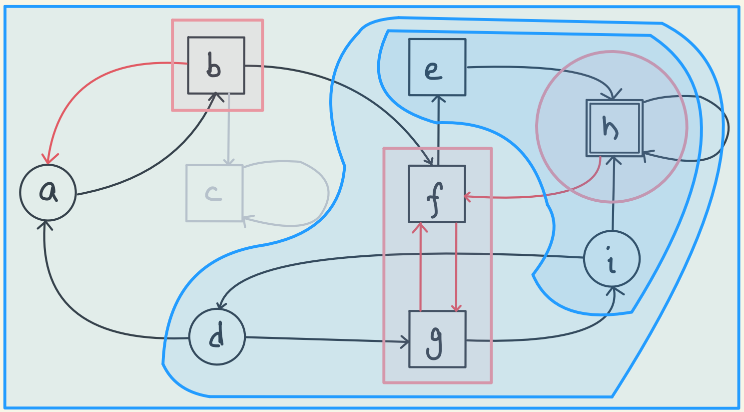
Need to restrict from going further away from $\boxed{h}$ eventually.

- Always not b → c.
- Eventually always not h → f
- Eventually always not f → g
- Eventually always not g → f
- Eventually always not b → a

colive templates

Computing Adequately Permissive Assumptions

coBüchi objective : Eventually always visit $\boxed{}$



Need to restrict from going further away from $\boxed{h}$ eventually.

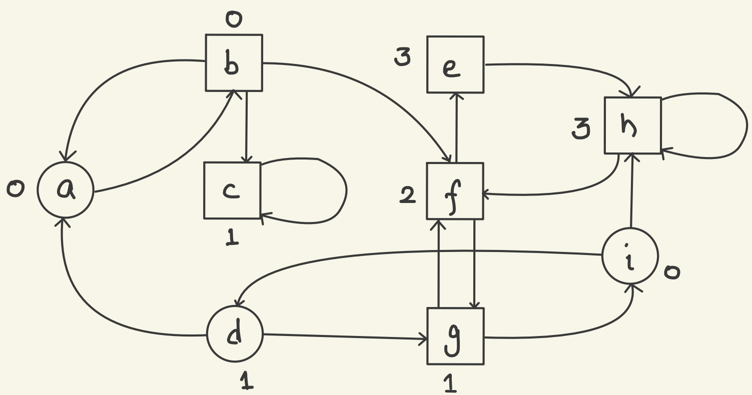
Runs in time $O(m+n)$

- Always not b → c.
- Eventually always not h → f
- Eventually always not f → g
- Eventually always not g → f
- Eventually always not b → a

colive templates

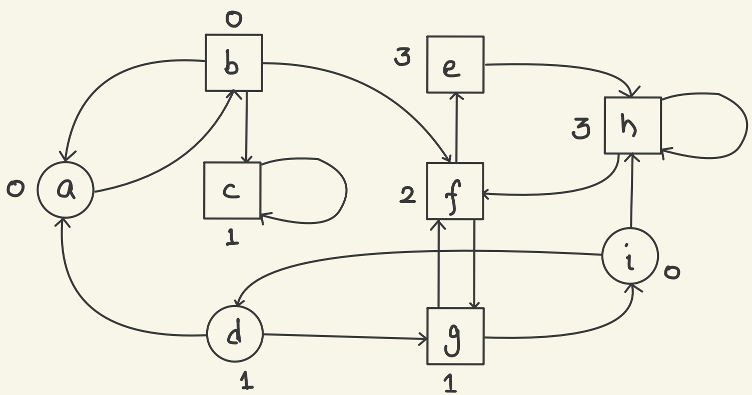
Computing Adequately Permissive Assumptions

Parity objective : highest priority visited infinitely is even



Computing Adequately Permissive Assumptions

Parity objective: highest priority visited infinitely is even

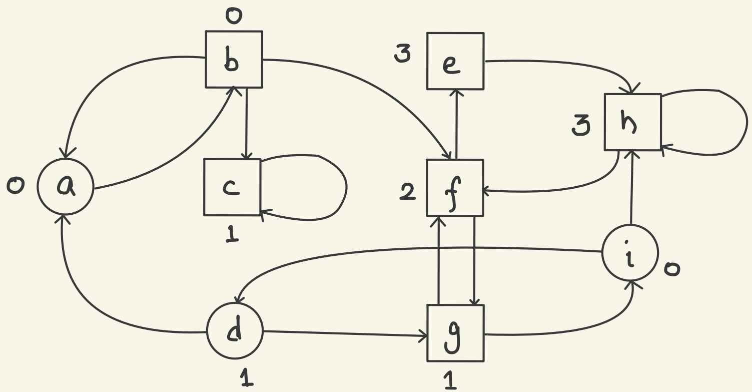


Needs conditional group liveness templates

$\square \lozenge C_3 \Rightarrow$ live group $(C_4 \cup C_6 \ldots)$

Computing Adequately Permissive Assumptions

Parity objective : highest priority visited infinitely is even



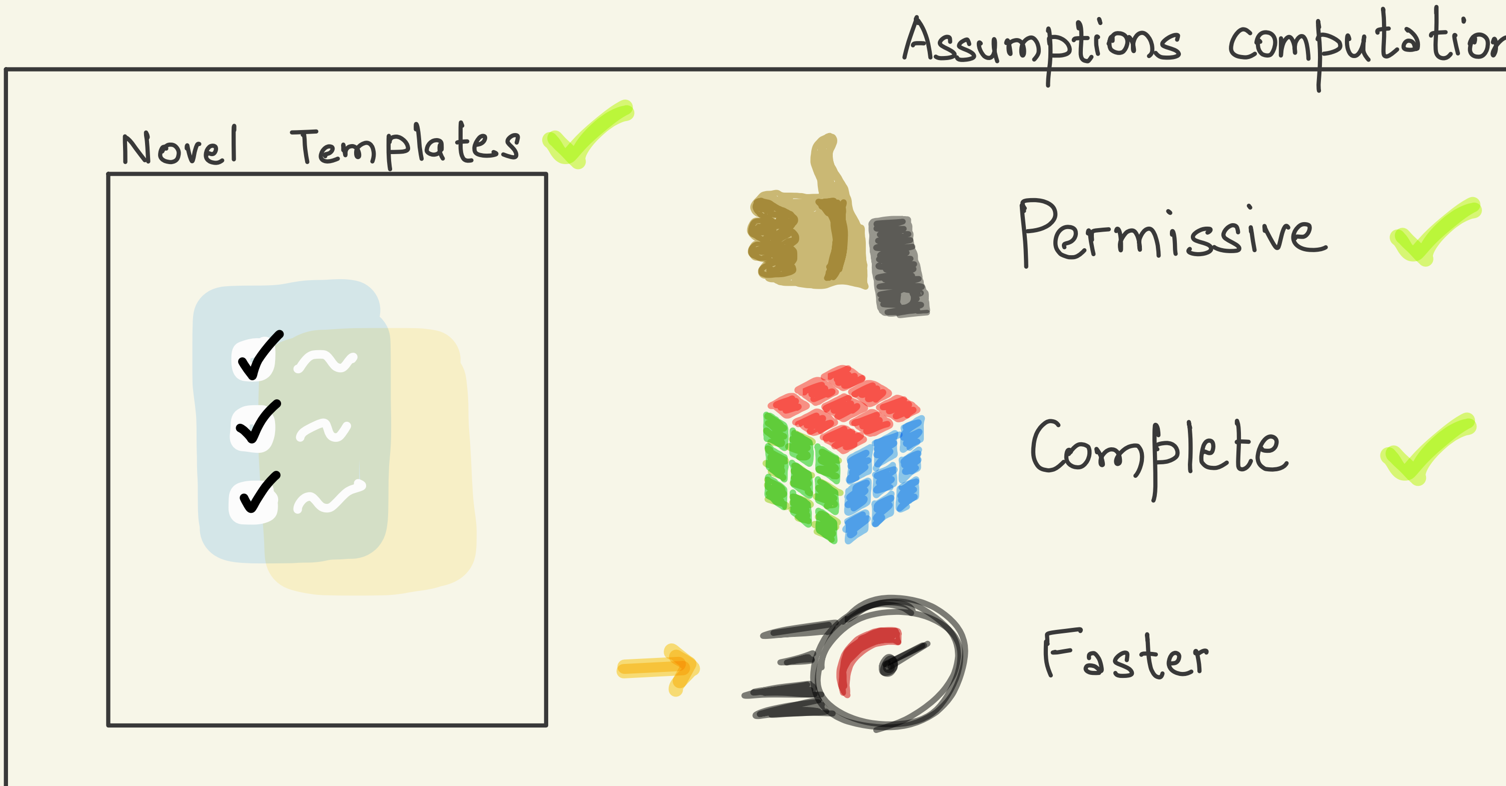
Runs in time $O(n^4)$

vertices

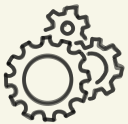
Needs conditional group liveness templates

$\square \lozenge C_3 \Rightarrow$ live group $(C_4 \cup C_6 ..)$

Checkpoint

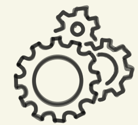

Assumptions computation
Novel Templates
Permissive
Complete
Faster

Experiments

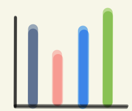


C++ Tool
SImPA

Experiments

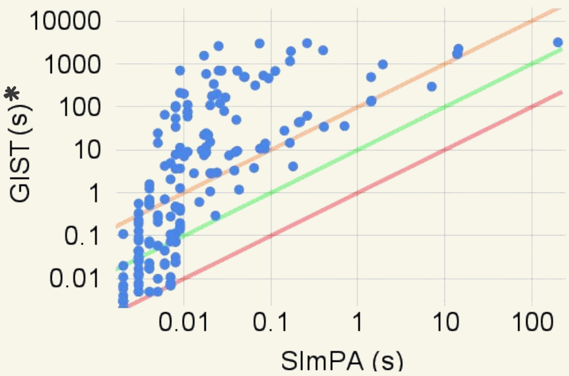


C++ Tool
SImPA

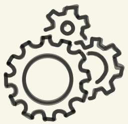


230 SYNTCOMP
benchmarks

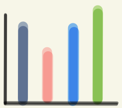
Experiments



* Chatterjee et. al, CAV '10

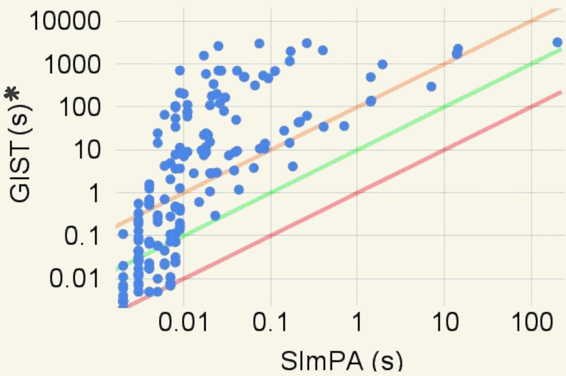


C++ Tool
SImPA



230 SYNTCOMP
benchmarks

Experiments



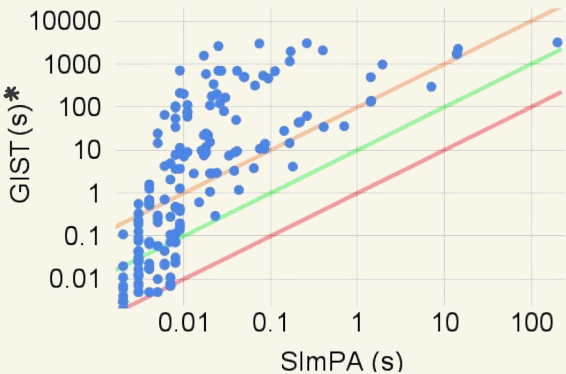
	SImPA	GIST*
Mean-time	64.8 s	1079.0 s
Non-timeout mean-time	64.8 s	209.2 s
Termination (1hr)	100%	74%
Generated assumptions	100%	91%
Fastest	100%	0%

* Chatterjee et. al, CAV '10

C++ Tool SImPA

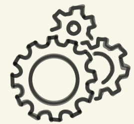
230 SYNTCOMP benchmarks

Experiments

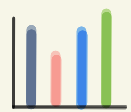


	SImPA	GIST*
Mean-time	64.8s	1079.0s
Non-timeout mean-time	64.8s	209.2s
Termination(1hr)	100%	74%
Generated assumptions	100%	91%
Fastest	100%	0%

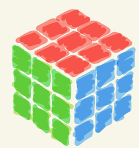
* Chatterjee et. al, CAV '10



C++ Tool
SImPA

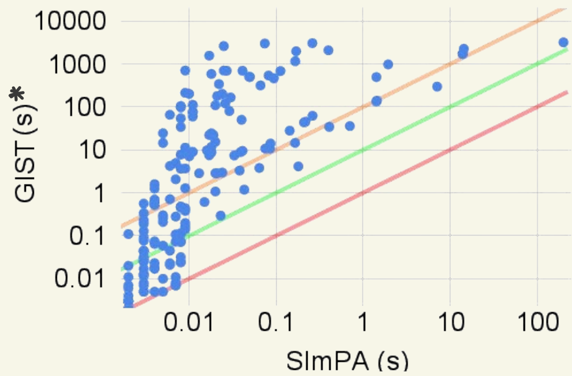


230 SYNTCOMP benchmarks



Always gives an assumption

Experiments



	SImPA	GIST*
Mean-time	64.8s	1079.0s
Non-timeout mean-time	64.8s	209.2s
Termination(1hr)	100%	74%
Generated assumptions	100%	91%
Fastest	100%	0%

* Chatterjee et. al, CAV '10

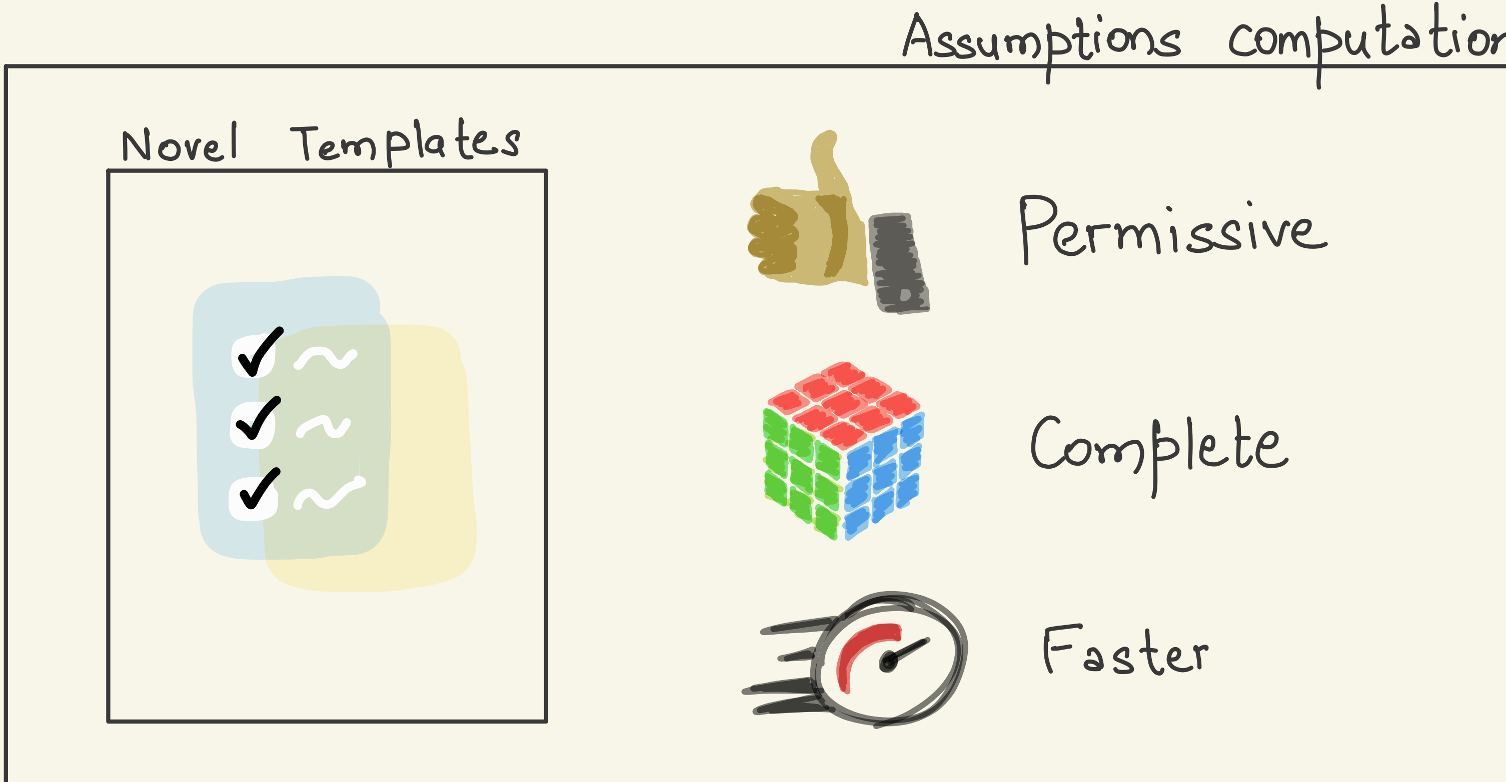
C++ Tool
SImPA

230 SYNTCOMP benchmarks

Always gives an assumption

Orders of magnitude faster

Summary


Assumptions computation
Novel Templates
Permissive
Complete
Faster

Assumptions on Environment

LTL formula φ on vertices of the game graph

Sufficient : $v \in \langle\langle s,e \rangle\rangle \phi \Rightarrow \exists \pi_s . \forall \pi_e$, $\pi_s \pi_e$ play $\models \varphi$

Assumptions on Environment

LTL formula φ on vertices of the game graph

Sufficient : $\underbrace{v \in \langle\langle s,e \rangle\rangle \phi}_{\text{cooperative winning region}} \Rightarrow \exists \underbrace{\pi_s}_{\text{system strategy}} . \forall \underbrace{\pi_e}_{\text{environment strategy}} , \underbrace{\mathcal{L}(\pi_s \pi_e, v)}_{\pi_s \pi_e \text{ play from } v} \models \varphi$

Assumptions on Environment

LTL formula φ on vertices of the game graph

following φ

Sufficient: $\underbrace{v \in \langle\langle s,e \rangle\rangle \varphi}_{\text{cooperative winning region}} \Rightarrow \exists \underbrace{\pi_s}_{\text{system strategy}} . \forall \underbrace{\pi_e}_{\text{environment strategy}} , \underbrace{h(\pi_s \pi_e, v)}_{\pi_s \pi_e \text{ play from } v} \models \varphi$

Assumptions on Environment

LTL formula ψ on vertices of the game graph

Sufficient : $v \in \langle\langle s,e \rangle\rangle \phi \Rightarrow \exists \pi_s . \forall \pi_e , \mathcal{L}(\pi_s \pi_e, v) \models \psi$

Implementable : $\exists \pi_e . \forall \pi_s . \mathcal{L}(\pi_s \pi_e) \models \psi$

Assumptions on Environment

LTL formula ψ on vertices of the game graph

Sufficient : $v \in \langle\langle s,e \rangle\rangle \phi \Rightarrow \exists \pi_s . \forall \pi_e , \mathcal{L}(\pi_s \pi_e, v) \models \varphi$

Implementable : $\exists \pi_e . \forall \pi_s . \mathcal{L}(\pi_s \pi_e) \models \psi$

Environment has a strategy to satisfy ψ from every vertex

Assumptions on Environment

LTL formula ψ on vertices of the game graph

Sufficient : $v \in \langle\langle s,e \rangle\rangle \phi \Rightarrow \exists \pi_s . \forall \pi_e , \mathcal{L}(\pi_s \pi_e, v) \models \varphi$

Implementable : $\exists \pi_e . \forall \pi_s . \mathcal{L}(\pi_s \pi_e) \models \psi$

Permissive : $\mathcal{L}(\phi) \subseteq \mathcal{L}(\psi)$

Assumptions on Environment

LTL formula ψ on vertices of the game graph

Sufficient : $v \in \langle\langle s,e \rangle\rangle \phi \Rightarrow \exists \pi_s . \forall \pi_e , \mathcal{L}(\pi_s \pi_e, v) \models \varphi$

Implementable : $\exists \pi_e . \forall \pi_s . \mathcal{L}(\pi_s \pi_e) \models \psi$

Permissive : $\mathcal{L}(\phi) \subseteq \mathcal{L}(\psi)$

Adequately permissive assumption