THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

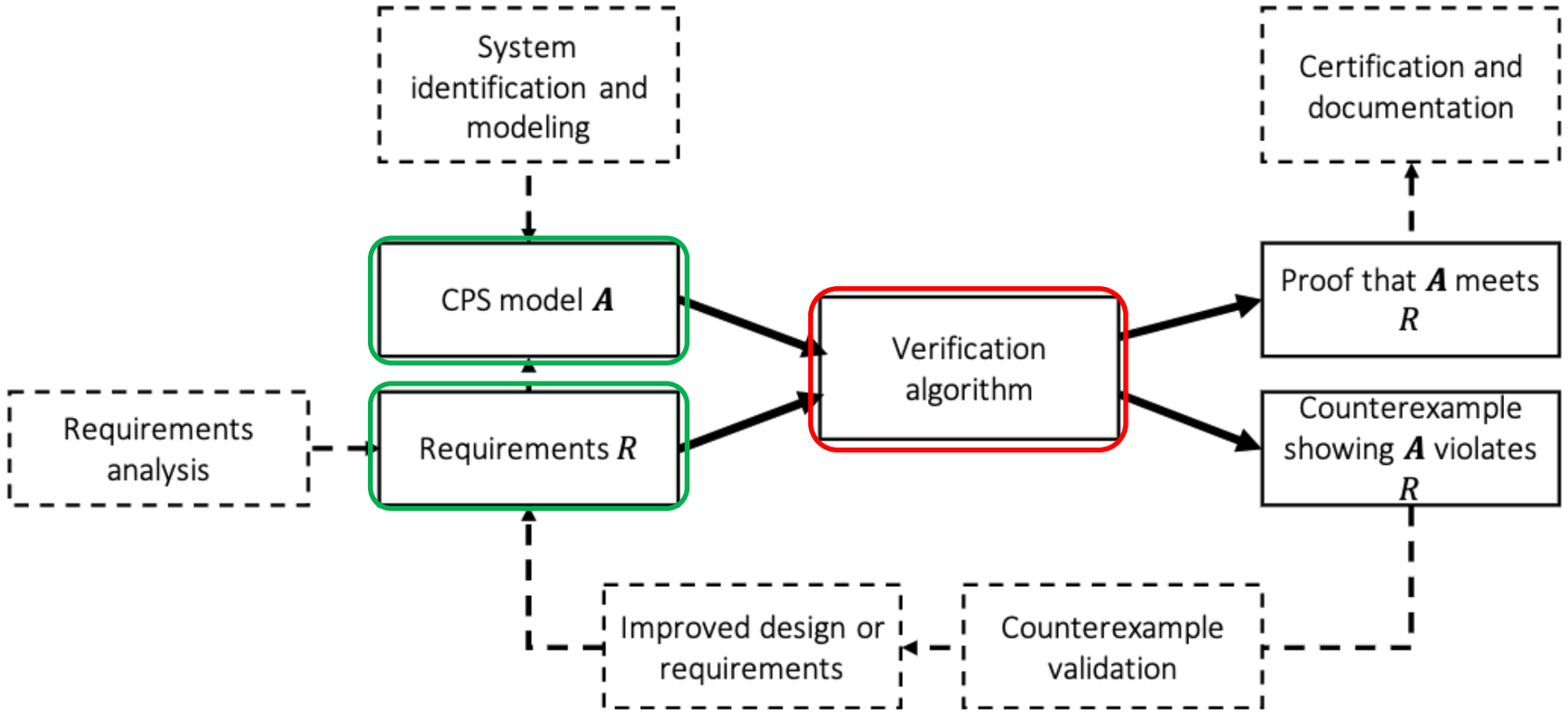
Verifying Cyber-Physical Systems

Chapter 8 – Abstractions and Compositional Reasoning

Abenezer Taye

Nov 14, 2022

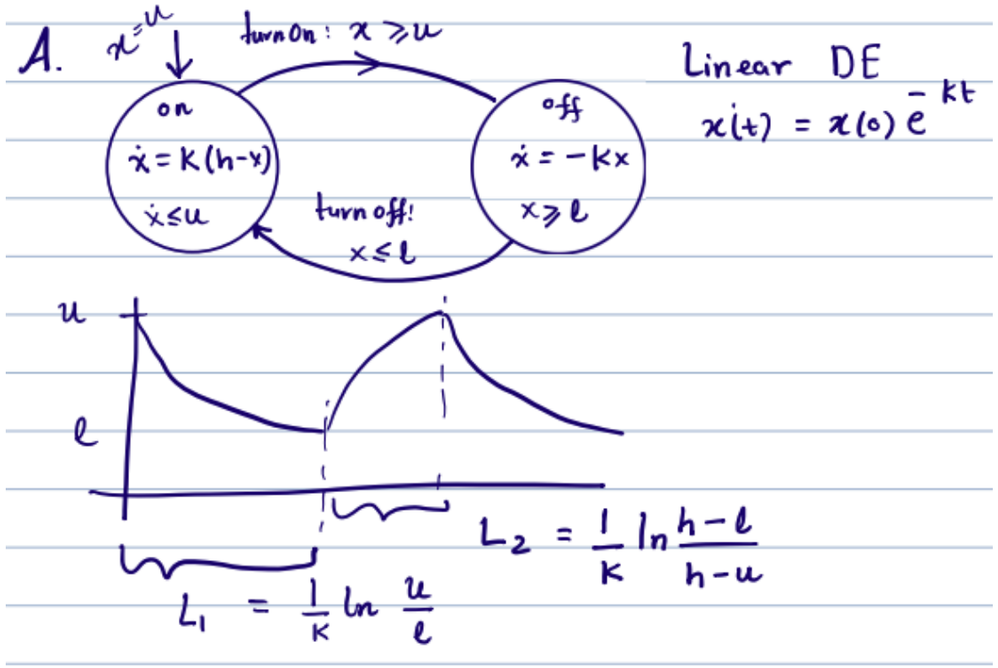
System Design Ecosystem



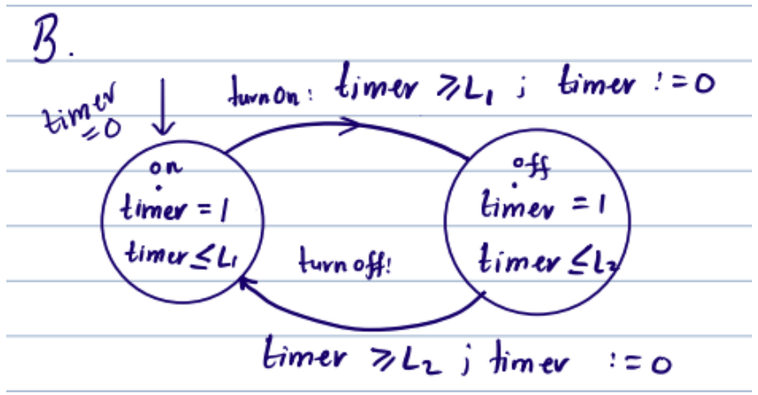
What is an Abstraction?

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

- Thermostat Automaton



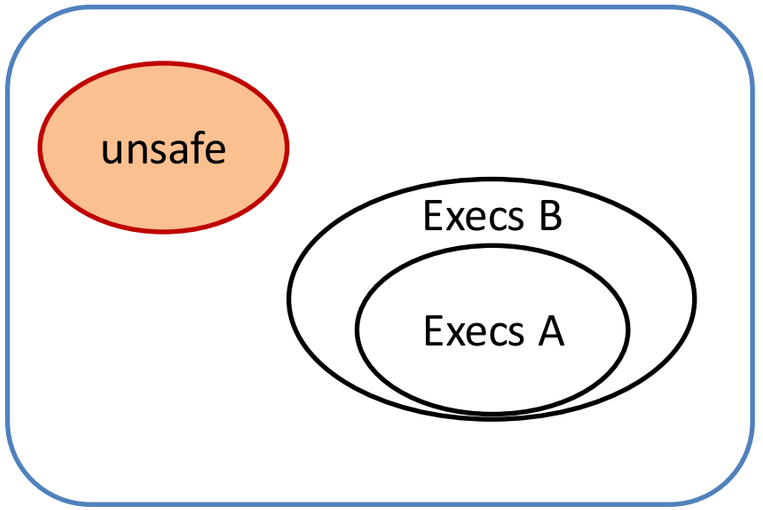
- If we only care about the timing behavior of the thermostat



What is an Abstraction?

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

- How can we show that B indeed has the “same” timing behavior as A?
- More generally, we may only care about certain aspects of A’s executions such as
  - Timing
  - Subset of continuous variables
  - Control state reachability, etc.
- How can we show that B is equivalent to A wrt the aspects of behavior we care about?
- How can we come up with an “equivalent” B that is simpler to analyze?
- Therefore, proving safety of B ➔ safety of A



Mathematically, we would like to show that $\forall \boldsymbol{\alpha} \in \boldsymbol{Execs_A}\ \exists\ \boldsymbol{\beta} \in \boldsymbol{Execs_B}\ \boldsymbol{such\ that}\ \boldsymbol{\alpha} = \boldsymbol{\beta}$

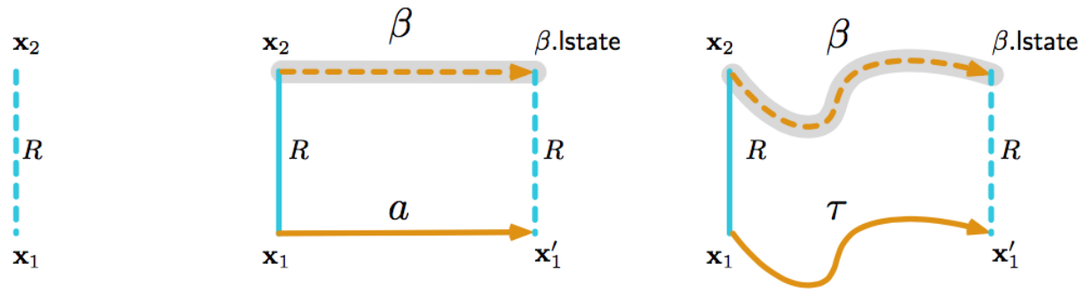
B is said to be an Abstraction of A

How to prove B abstracts A?

Forward simulation relation from $\mathcal{A}_1$ **to** $\mathcal{A}_2$ is a relation R $\subseteq val(X_1) \times val(X_2)$ such that

1. For every $\mathbf{x_1} \in \Theta_1$ there exists $\mathbf{x_2} \in \Theta_2$ such that $\mathbf{x_1}$ R $\mathbf{x_2}$
2. For every $\mathbf{x_1} \rightarrow_{a_1} \mathbf{x_1'} \in \mathcal{D}$ and $\mathbf{x_2}$ such that $\mathbf{x_1}$ R $\mathbf{x_2}$, there exists $\mathbf{x_2'}$ such that
   - $\mathbf{x_2} \rightarrow_{a_1} \mathbf{x_2'}$ and
   - $\mathbf{x_1'}$ R $\mathbf{x_2'}$
3. For every $\boldsymbol{\tau_1} \in \mathcal{T}_1$ and $\mathbf{x_2}$ such that $\tau_1.fstate$ R $\mathbf{x_2}$, there exists $\tau_2 \in \mathcal{T}_2$ that
   - $\mathbf{x_2} = \tau_2.fstate$ and
   - $\mathbf{x_1'}$ R $\tau_2.lstate$
   - $\tau_2.\mathrm{dom} = \tau_1.dom$

Theorem. If there exists a forward simulation relation from hybrid automaton $\mathcal{A}_1$ to $\mathcal{A}_2$ then for every execution of $\mathcal{A}_1$ there exists a corresponding execution of $\mathcal{A}_2$.

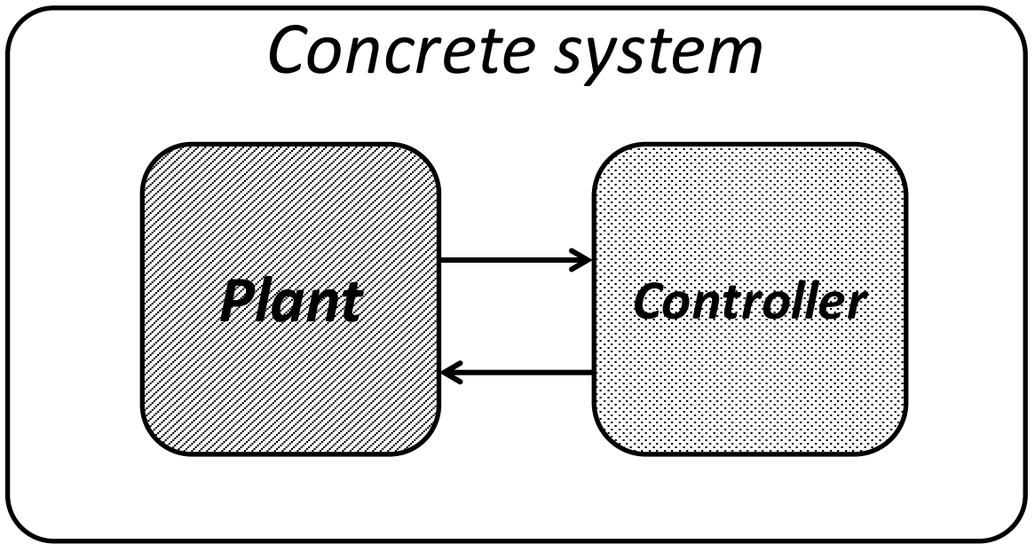


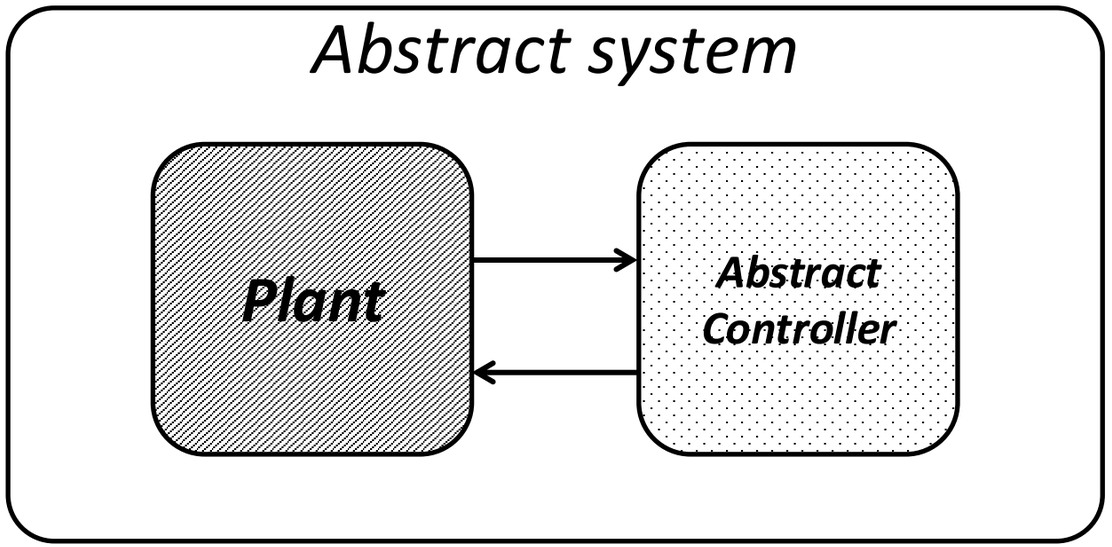
Properties of forward simulation

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

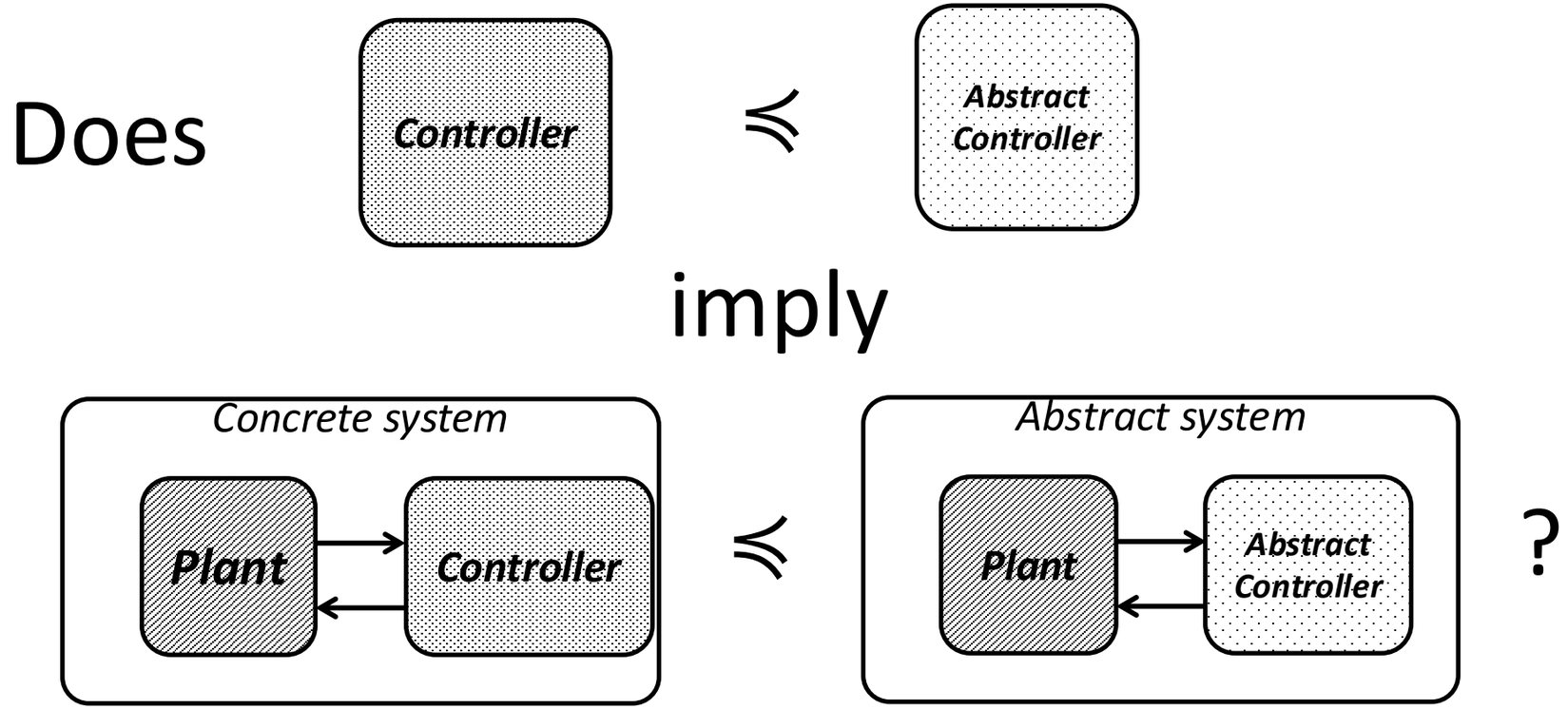
- Different types of simulation relations (R) depending on different notions for "Trace"
  - Match for all variable values, action names, and time duration of trajectories **(abstraction)**
  - Match variables but not time **(time abstract simulation)**
  - Match a subset (external) of variables and actions **(trace inclusion)**
  - Match single action/trajectory of A with a sequence of actions and trajectories of B
- Comparable automata → automata with matching sets of variables and actions
- Let $\mathcal{A}, \mathcal{B}, \text{and } \mathcal{C}$ be comparable HAs. If R_1 is a forward simulation from $\mathcal{A}$ to $\mathcal{B}$ and R_2 is a forward simulation from $\mathcal{B}$ to $\mathcal{C}$, then **$R_1 \circ R_2$ is a forward simulation from $\mathcal{A}$ to $\mathcal{C}$ ($\mathcal{A}$ implements $\mathcal{C}$)**
- If R is a forward simulation from $\mathcal{A}$ to $\mathcal{B}$ and R^{-1} is a forward simulation from $\mathcal{B}$ to $\mathcal{A}$ then R is called a **bisimulation** and $\mathcal{B}$ and $\mathcal{A}$ are **bisimilar**
- Bisimilarity is an **equivalence relation**

Substituting with abstractions





Substituting with abstractions



Substitutivity

Theorem: Suppose $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{B}$ have the same external interface and $\mathcal{A}_1$, $\mathcal{A}_2$ are compatible with $\mathcal{B}$.
If $\mathcal{A}_1 \preccurlyeq \mathcal{A}_2$ then $\mathcal{A}_1 || \mathcal{B} \preccurlyeq \mathcal{A}_2 || \mathcal{B}$

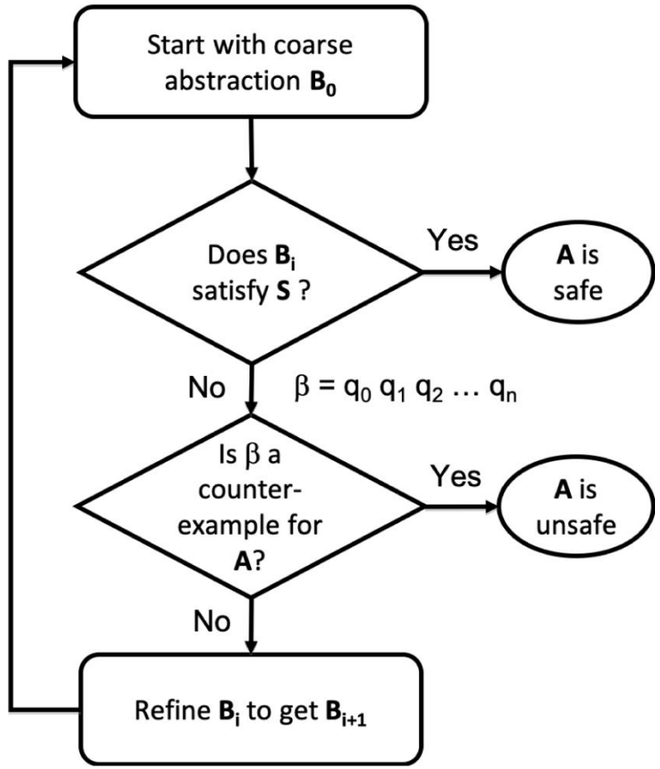
Stronger substitutivity result

Theorem: $\mathcal{A}_1 || \mathcal{B}_2 \preccurlyeq \mathcal{A}_2 || \mathcal{B}_2$ and $\mathcal{B}_1 \preccurlyeq \mathcal{B}_2$ then $\mathcal{A}_1 || \mathcal{B}_1 \preccurlyeq \mathcal{A}_2 || \mathcal{B}_2$.

Counterexample guided abstraction refinement (CEGAR)

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

- A general algorithmic framework for automatically constructing and verifying property-specific abstractions [Clarke:2000]
- CEGAR has been applied to discrete automata, software, and hybrid systems [Holzman 00,Ball 01, Alur 2006,Clarke 2003, Fehnker2005, Prabhakar 15, Roohi 17]
- The goal is to verify a requirement S for a system or an automaton A



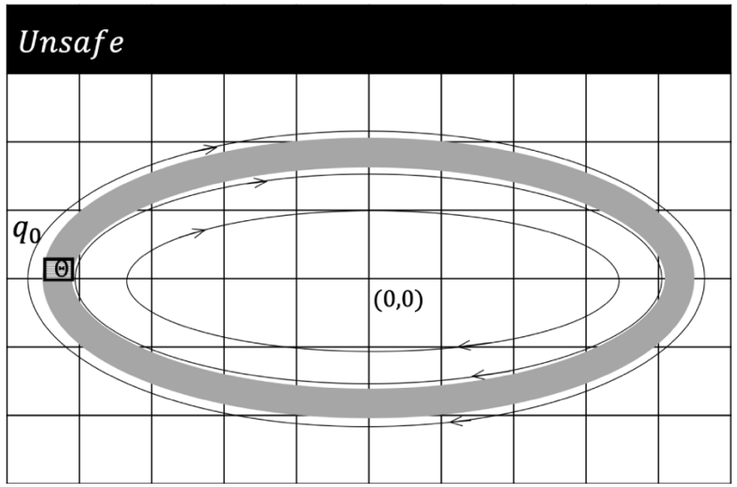
Basic CEGAR loop for verifying a requirement S for a system A

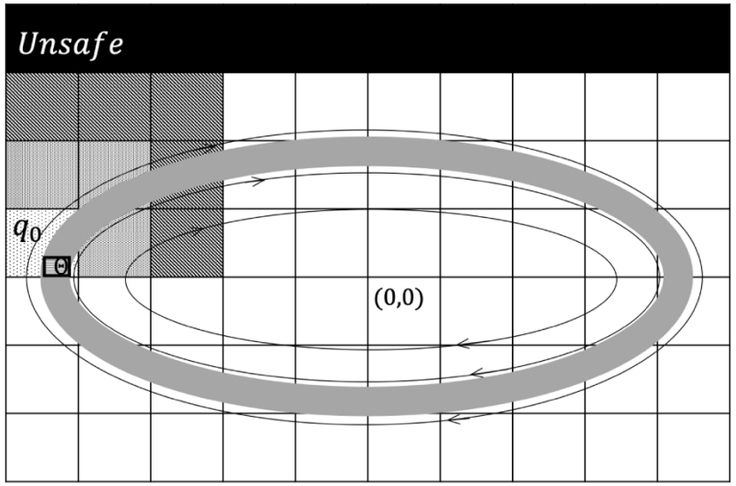
Key design choices in CEGAR

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

- Space of the abstract automata (finite, timed, linear)
- Model checker for abstract automaton
- Counter-example validation procedure
- Refinement strategy

Space of the abstract automata





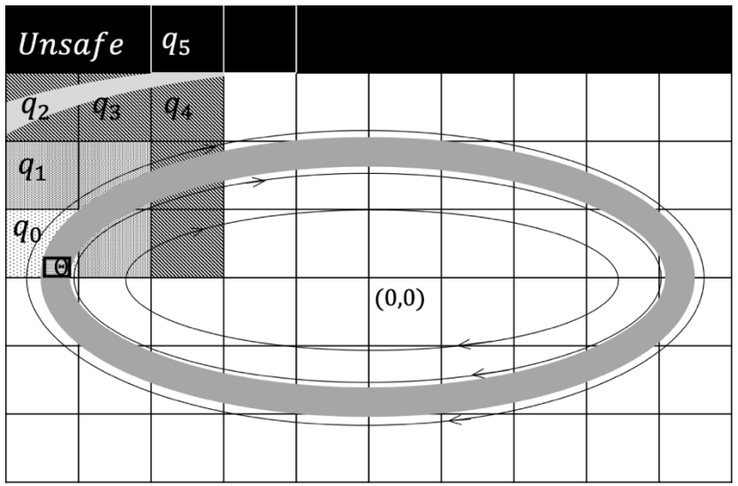
Model checker

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

- Model checker takes an input abstract automaton B and **decides whether B meets the requirement S**
- If B violates S, the model checker has to produce a counterexample, β.
- The type of requirement S and the choice of the type of abstract automaton determine the type of model checker we can employ
  - If B is timed automaton → UPPAL (scales exponentially with the number of clock variables)
  - If B is an RHA → model checking is technically undecidable but tools like HyTech and PHAVer can be used without the guarantee of termination

Counterexample validation

- **Validation step (step 2)** in CEGAR loop takes an abstract counterexample β and checks whether there is a corresponding execution α of the concrete automaton A.
- For example, the counterexample for the discrete automaton B discovered by the discrete model checker is the execution $\beta = q_0, q_1, q_2, q_3, q_4, q_5$
- Now, we have to check whether there is an execution of A that traverses those regions.
- One way to implement that is to compute the set of states that are backward-reachable from $R^{-1}(q_5)$ and that touch states in β, and check whether this set intersects with q_0



Refinement strategy

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

- Final step in CEGAR loop refines an abstract automaton B_i to a new finer abstraction B_{i+1}
- Strategies for refinement include:
  - Splitting of a single location into multiple locations
  - Addition of variables
  - Splitting of transitions
  - Addition of higher-order terms in approximating nonlinear dynamics

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Thank You!!!

Q&A