Monitoring Off-the-Shelf Components

Min Zhou

mzhou@cs.uic.edu

University of Illinois at Chicago

joint work with Prasad Sistla and Lenore Zuck

Problem Description

C: off-the-shelf component

Φ_C: spec of C Φ_G : goal/user spec

If $(\Phi_C \rightarrow \Phi_G)$, then C can be used

Problem Description

C: off-the-shelf component

Φ_C: spec of C Φ_G : goal/user spec

If $(\Phi_C \rightarrow \Phi_G)$, then C can be used

Unfortunately, often this is not the case!

Problem Description

Unfortunately, often this is not the case!

- Why?

Problem Description

Unfortunately, often this is not the case!

- Why?
  - mismatch or under-specification of C

Problem Description

Unfortunately, often this is not the case!

- Why?
  - mismatch or under-specification of C
- Idea: construct a monitor M s.t.

$$(\Phi_C \wedge \Phi_M) \rightarrow \Phi_G$$

Problem Description

Unfortunately, often this is not the case!

- Why?
  - mismatch or under-specification of C
- Idea: construct a monitor M s.t.

$$(\Phi_C \wedge \Phi_M) \rightarrow \Phi_G$$

- M monitors the executions of C for violations of Φ_M

Requirements on M

Requirements on M

- Φ_M: safety property

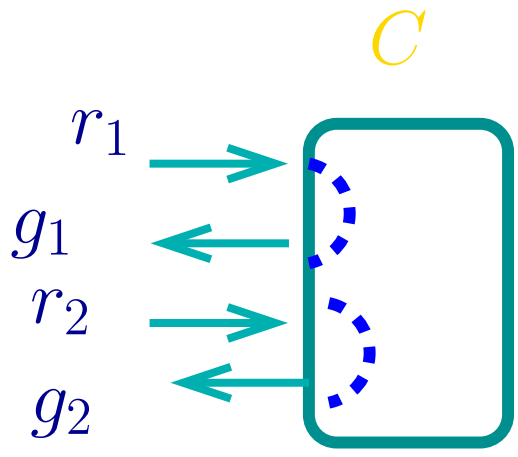
Requirements on M

- Φ_M: safety property
- $(\Phi_C \wedge \Phi_M) \rightarrow \Phi_G$, i.e., $\Phi_M \rightarrow (\neg\Phi_C \vee \Phi_G)$

Requirements on M

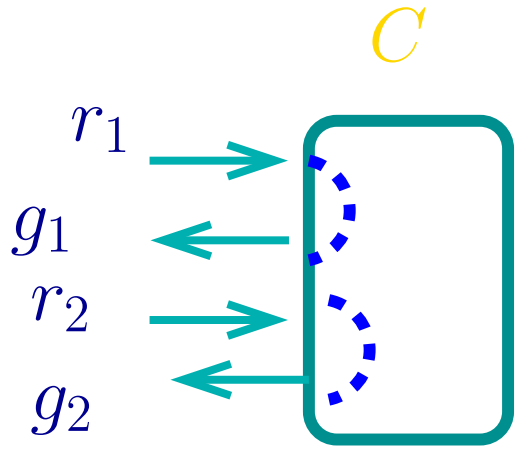
- Φ_M: safety property
- $(\Phi_C \wedge \Phi_M) \rightarrow \Phi_G$, i.e., $\Phi_M \rightarrow (\neg\Phi_C \vee \Phi_G)$
- Φ_M: as "general" as possible

Example: Permission Manager



$$\Phi_C : \wedge_{i=1}^{2} \Box(r_i \rightarrow \Diamond g_i)$$

Example: Permission Manager

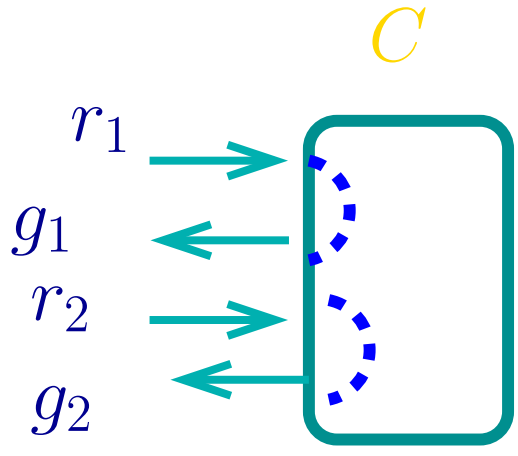


$\Phi_C : \wedge_{i=1}^{2} \Box(r_i \rightarrow \Diamond g_i)$

- Φ_G: priority for r_1

  $\Box(r_1 \rightarrow (\neg g_2 \ until \ g_1))$

Example: Permission Manager



$\Phi_C : \wedge_{i=1}^{2} \Box(r_i \rightarrow \Diamond g_i)$

- Φ_G: priority for r_1

  $\Box(r_1 \rightarrow (\neg g_2 \ until \ g_1))$

- Φ_M: $\Box(r_1 \rightarrow (\neg g_2 \ unless \ g_1))$

Previous Results

- Theorem [MSSZ05]: Let $\mathcal{A}$ be a Büchi, assume $L(\mathcal{A})$ is not a safety. Then for every safety $L' \subset L(\mathcal{A})$, there exists a safety L'' s.t.

$$L' \subset L'' \subset L(\mathcal{A})$$

Previous Results

- Theorem [MSSZ05]: Let $\mathcal{A}$ be a Büchi, assume $L(\mathcal{A})$ is not a safety. Then for every safety $L' \subset L(\mathcal{A})$, there exists a safety L'' s.t.

$$L' \subset L'' \subset L(\mathcal{A})$$

- Generally, there is no maximal Φ_M s.t.

$$\Phi_M \rightarrow (\neg\Phi_C \vee \Phi_G)$$

Results

- Synthesizing increasingly larger Φ_M's from FSAs

Results

- Synthesizing increasingly larger Φ_M's from FSAs
- Synthesizing of monitors directly from LTL temporal formulae

Results

- Synthesizing increasingly larger Φ_M's from FSAs
- Synthesizing of monitors directly from LTL temporal formulae
- Soundness and Completeness

Simple Solution

- Given: $\mathcal{A}$ - Büchi automaton (NBA)

  k - positive integer

Simple Solution

- Given: $\mathcal{A}$ - Büchi automaton (NBA)

  k - positive integer

- construct a NBA $\mathcal{B}_k$

Simple Solution

- Given: $\mathcal{A}$ - Büchi automaton (NBA)

  k - positive integer

- construct a NBA $\mathcal{B}_k$
  - $L(\mathcal{B}_k)$ is a safety

Simple Solution

- Given: $\mathcal{A}$ - Büchi automaton (NBA)

  k - positive integer

- construct a NBA $\mathcal{B}_k$
  - $L(\mathcal{B}_k)$ is a safety
  - $L(\mathcal{B}_k) \subseteq L(\mathcal{A})$

Simple Solution

- Given: $\mathcal{A}$ - Büchi automaton (NBA)

  k - positive integer

- construct a NBA $\mathcal{B}_k$
  - $L(\mathcal{B}_k)$ is a safety
  - $L(\mathcal{B}_k) \subseteq L(\mathcal{A})$
  - $L(\mathcal{B}_k) \subseteq L(\mathcal{B}_{k+1})$

Simple Solution

- Given: $\mathcal{A}$ - Büchi automaton (NBA)

  k - positive integer

- construct a NBA $\mathcal{B}_k$
  - $L(\mathcal{B}_k)$ is a safety
  - $L(\mathcal{B}_k) \subseteq L(\mathcal{A})$
  - $L(\mathcal{B}_k) \subseteq L(\mathcal{B}_{k+1})$
- How?

How

- Use a modulo-k counter, init = $k - 1$

How

- Use a modulo-k counter, init = $k - 1$
- For transitions from non-accepting states, decrement the counter

How

- Use a modulo-k counter, init = $k - 1$
- For transitions from non-accepting states, decrement the counter
- For transitions from accepting states, reset the counter

Sound but Incomplete

- $L(\mathcal{A}) = \Box \Diamond P$
- $S = (P, \bar{P}, P, (\bar{P})^2, P, (\bar{P})^3, \cdots)$
- No $\mathcal{B}_k$ recognizes S

Bounded Automata

- Intuition: Make the reset value of counter depends on the past

Bounded Automata

- Intuition: Make the reset value of counter depends on the past
- Each state: (r, q, i) where

Bounded Automata

- Intuition: Make the reset value of counter depends on the past
- Each state: (r, q, i) where
  - r: is a "history" variable

Bounded Automata

- Intuition: Make the reset value of counter depends on the past
- Each state: (r, q, i) where

  r: is a "history" variable

  $q \in Q_A$

Bounded Automata

- Intuition: Make the reset value of counter depends on the past
- Each state: (r, q, i) where

  r: is a "history" variable

  $q \in Q_A$

  i: is the counter, possibly ∞ (non accepting)

Bounded Automata

- Intuition: Make the reset value of counter depends on the past
- Each state: (r, q, i) where

  r: is a "history" variable

  $q \in Q_A$

  i: is the counter, possibly ∞ (non accepting)

- For transitions from non-accepting states, decrease the counter or set it to ∞ (where it remains)

Soundness and Partial Completeness

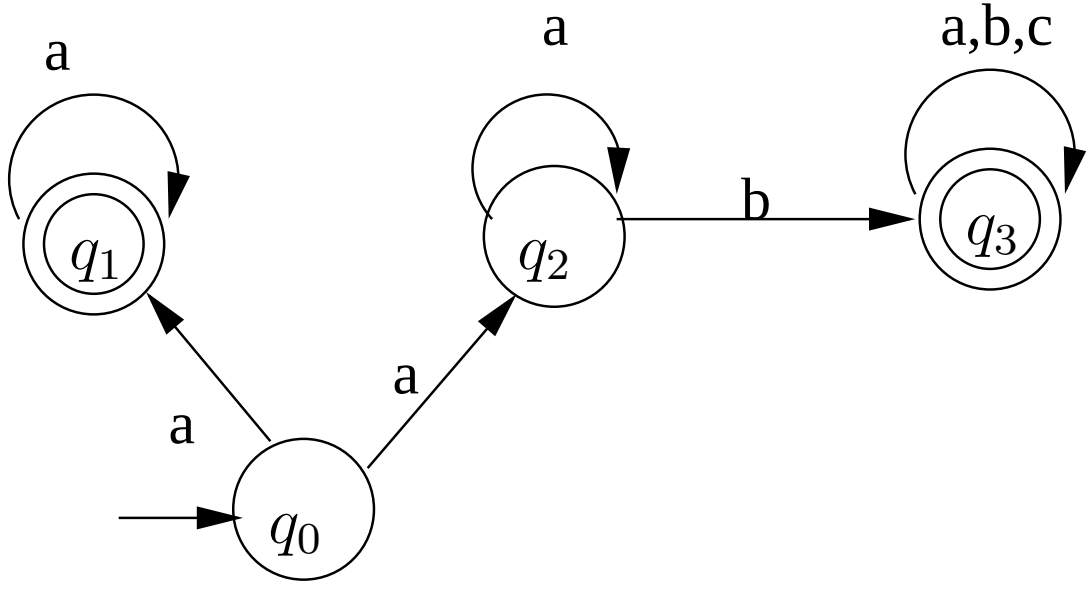
Given an automaton $\mathcal{A}$, a bounded automaton $\mathcal{N}$ over it:

- $L(\mathcal{N}) \subseteq L(\mathcal{A})$
- $L(\mathcal{N})$ is a safety
- If $\mathcal{A}$ is deterministic Büchi, $S \subseteq L(\mathcal{A})$ is safety, then there exists some bounded automaton $\mathcal{N}$ that accepts S

Why partial

$L = \{a^i b \Sigma^\omega \ : i > 0\} \cup \{a^\omega\}$

Cannot get L, only $\{a^i b \Sigma^\omega \ : 0 < i \leq k\} \cup \{a^\omega\}$ where k is the initial value of counter



Streett Automata

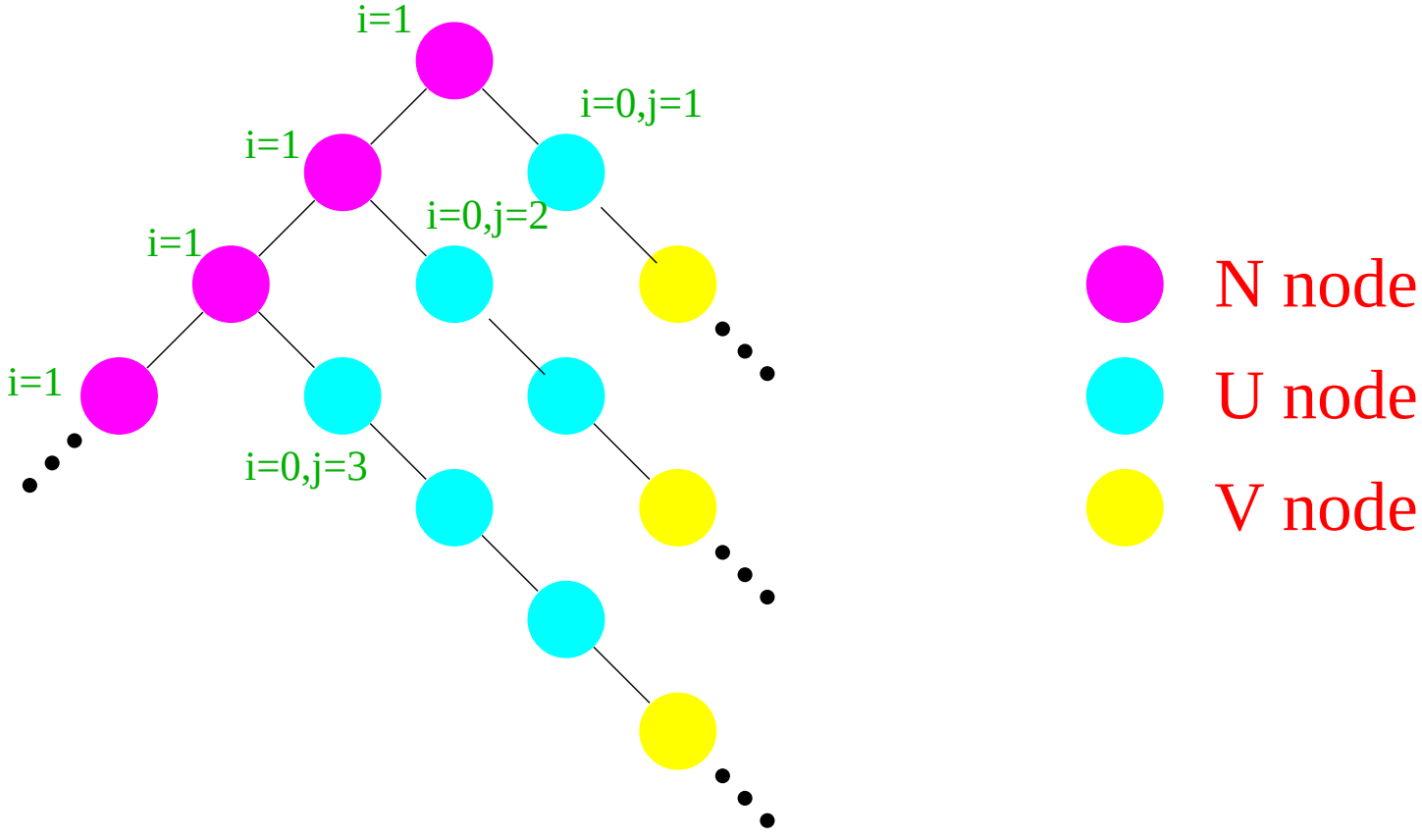
- Accepting condition: set of pairs (U, V), $U, V \subseteq Q$
- A run is accepting if, for every pair (U, V), $\Box\Diamond U \Rightarrow \Box\Diamond V$
- NBAs can be converted into equivalent deterministic Streetts [Safra88]

Extended Bounded Automata(EBA)

- Use 2 counters i, j for every (U, V) pair
- Transitions ending in U nodes
  - $i > 0$: decrease i
  - $i = 0$: i remains 0, decrease j
- Transitions ending in neither U nor V nodes
  - $i > 0$: non-increase i
  - $i = 0$, i remains 0, non-increase j

Why two counters

- N nodes on left path
- Unbounded U nodes on right paths



Soundness and Completeness

- For every EBA $\mathcal{E}$ over $\mathcal{A}$, $L(\mathcal{E})$ is a safety and $L(\mathcal{E}) \subseteq L(\mathcal{A})$
- Let $\mathcal{A}$ be a deterministic Streett, for each safety $S \subseteq L(\mathcal{A})$, there exists an EBA $\mathcal{B}$ that accepts S

Synthesizing from Formulae

- Semantic Approach
  - Associate a counter for each $\mathcal{U}$ subformula in the construction of tableau
  - A counter can be active or inactive
  - A transition is disabled if the counter is 0

Syntactic Approach

- Push negations to propositions
- Replace every $\mathcal{U}$ with $\mathcal{U}_{<k}$

  ($\mathcal{U}_{<k}$: bounded until operator)
- For a temporal formula ϕ and a positive integer k, ϕ_k is a safety property which implies ϕ

Related Work

- Pnueli, Zaks, Zuck 2005(RV05) workshop: use game theoretic approach

- Larsen 90,91: Give methods for synthesizing a safety property with respect to a context

Conclusion and Future Work

- Methods for customizing off-the-shelf components by monitors

- Future work
  - Study the complexity of monitors
  - Develop real-time monitors
  - Construct active monitors
  - Consider more realistic examples