Closure Under Stuttering

References

- D. Paun, M. Chechik, B. Biechelle, "Production Cell Revisited", in Proceedings of SPIN'98, November 1998.
- D. Paun, M. Chechik, "Events in Linear-Time Properties", in Proceedings of International Symposium on Requirements Engineering, June 1999.
- M. Chechik, D. Paun, "Events in Property Patterns", in Proceedings of SPIN'99, September 1999.
- D. Paun, "On Closure Under Stuttering", M.S. Thesis, University of Toronto, Department of Computer Science, May 1999.

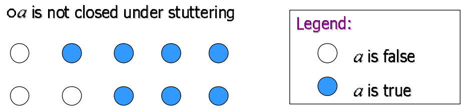
Closure Under Stuttering

Desired property of LTL formulas is *closure under stuttering* : interpretation of the formula remains the same under state sequences that differ only by repeated states [Abadi,Lamport'91].

- Guaranteed [Lamport'94] for a subset of LTL without the $\circ$ operator

Examples:

- $\Box a$ is closed under stuttering
- $\circ a$ is not closed under stuttering



Notation: <<F>> - F is closed under stuttering

Using LTL to Specify Production Cell System

- Case study initiated by Forchrungszentrum Informatik (FZI)
- Aimed to show applicability of formal methods to real-world examples

Example property:

The magnet of the crane may be deactivated only when the magnet is above the feedbelt.

Resulting LTL formula:

$\Box((activate \wedge \circ\neg activate) \Rightarrow \circ(head\_ver = DOWN))$

Is this formula closed under stuttering?!!

Related Work

- Determining whether an arbitrary LTL formula is closed under stutterung is PSPACE-complete [Peled,Wilke,Wolper'96]
  - Tableu-based, $$$ approach
- A computationally-feasible algorithm for determining closure under stuttering for a subclass of formulas has been proposed [Holzmann,Kupferman'96] but not implemented in SPIN
  - Algorithm cannot be applied by hand
  - How useful in practice?

Our goal:

- Want to have syntactical restrictions on LTL (like "no next state") that guarantee that the resulting formula is closed under stuttering
- Want the approach to apply to real-life problems

Edges

$\Box((activate \wedge \circ\neg activate) \Rightarrow \circ(head\_ver = DOWN))$

an *edge* (a change of value)

Formally, if A is an LTL formula, then

$\uparrow A = \neg A \wedge \circ A$ -- up or rising edge

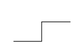
$\downarrow A = A \wedge \circ\neg A$ -- down or falling edge

$\updownarrow A = \uparrow A \vee \downarrow A$ -- any edge

Example: $\uparrow\Box A$

Edges ≈ events

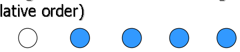
(Logical) edges ≈ signal edges



Main Result

Observation:

stuttering does not add or delete edges (or change their relative order)



Theorem:

$<<A>> \wedge <<B>> \Rightarrow << \Diamond(\neg A \wedge \circ A \wedge \circ B)>>$

Proof: in [Paun99]

Some Properties of Edges

- Edges are related:
  - $\uparrow\neg A = \downarrow A$
  - $\downarrow\neg A = \uparrow A$
  - $\updownarrow\neg A = \updownarrow A$
- Edges interact with each other:
  - $\downarrow\downarrow A = \downarrow A$
  - $\uparrow\downarrow A = \downarrow\downarrow A$
- Edges interact with boolean operators:
  - $\uparrow(A \wedge B) = (\uparrow A \wedge \circ B) \vee (\uparrow B \wedge \circ A)$
- Edges interact with temporal operators
  - $\uparrow\circ A = \circ\uparrow A$
  - $\downarrow\Box A = false$
  - $\downarrow\Diamond A = \downarrow A \wedge \circ\Box\neg A$
  - $\uparrow(A\, U\, B) = \neg(A \vee B) \wedge \circ(A\, U\, B)$

Some Properties of Closure Under Stuttering

a is a variable or a constant $\Rightarrow <<a>>$

$<<A>> = <<\neg A>>$

$<<A>> \wedge <<B>> \Rightarrow <<A \wedge B>>$

$<<A>> \Rightarrow <<\Box A>>$

$<<A>> \Rightarrow << \Diamond A>>$

$<<A>> \wedge <<B>> \Rightarrow <<A\, U\, B>>$

$<<A>> \wedge <<B>> \Rightarrow <<A * B>>$, where $* \in \{\wedge, \vee, \Rightarrow, \Leftarrow, =\}$

Formulas of the form $<<A>> \Rightarrow$ f ($\updownarrow A$): edges $\uparrow$ and $\downarrow$ can be used interchangeably.

Closure Under Stuttering Properties

Property 1 (Existence)

$<<A>> \wedge <<B>> \wedge <<C>> \Rightarrow << \Diamond(\uparrow A \wedge \circ B \wedge C)>>$

with simplified versions:

$<<A>> \wedge <<B>> \Rightarrow << \Diamond(\uparrow A \wedge B)>>$

$<<A>> \wedge <<B>> \Rightarrow << \Diamond(\uparrow A \wedge \circ B)>>$

Property 2 (Universality)

$<<A>> \wedge <<B>> \wedge <<C>> \Rightarrow << \Box(\uparrow A \Rightarrow (\circ B \vee C))>>$

with simplified versions:

$<<A>> \wedge <<B>> \Rightarrow << \Box(\uparrow A \Rightarrow B)>>$

$<<A>> \wedge <<B>> \Rightarrow << \Box(\uparrow A \Rightarrow \circ B)>>$

Closure Under Stuttering Properties (Cont'd)

Property 3 (Until)

$<<A>> \wedge <<B>> \wedge <<C>> \wedge <<D>> \wedge <<E>> \wedge <<F>>$

$\Rightarrow << (\neg\uparrow A \vee \circ B \vee C)\, U\, (\uparrow D \wedge \circ E \wedge F)>>$

with many simplified versions.

Examples:

The magnet of the crane may be deactivated only when the magnet is above the feedbelt.

$\Box(\downarrow activate \Rightarrow \circ(head\_ver = DOWN))$

Initially, no items should be dropped on the table before the operator pushes and releases the GO button

$\neg\downarrow hold\; U \downarrow button$

Quick Summary

- We introduced the notion of edges for LTL
- We provided a set of theorems that enable syntax-based analysis of a large class of formulas for closure under stuttering.
- Such theorems can be added to a theorem-prover for mechanized checking.

!! But the language of edges is not closed !!

Example: $\updownarrow A$

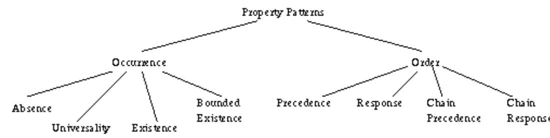
Are the properties that can be identified using our method useful in practice?

Application: Property Patterns

- Pattern-based approach [Dwyer,Avrunin,Corbett'98,'99]
  - Presentation, codification and reuse of property specifications
  - Easy conversion between formalisms: CTL, LTL, QRE, GIL...
  - Goal: to enable novice users to express complex properties effectively
    - LTL properties are state-based
- Apply our theory to
  - extend the pattern-system with events for LTL properties
  - check closure-under-stuttering of resulting formulas

13

Pattern Hierarchy

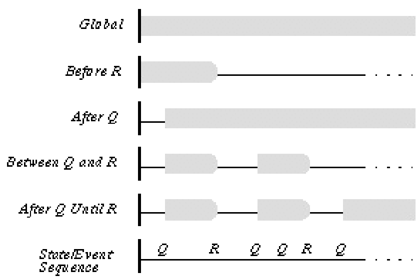


- Absence — A condition does not occur within a scope
- Existence — A condition must occur within a scope
- Universality — A condition occurs throughout a scope
- Response — A condition must always be followed by another within a scope
- Precedence — A condition must always be preceded by another within a scope.

14

Scopes

Scopes are regions of interest over which the condition is evaluated.



15

Example

LTL formulation of the property

S precedes P between Q and R

(Precedence pattern with "between Q and R" scope) is

$$\Box((Q \wedge \Diamond R) \Rightarrow (\neg P \; U \; (S \vee R)))$$

Note that S, P, Q, R are states.

16

Extending the Pattern System

- Want to extend LTL patterns to reasoning about events
- "next" operator: are resulting properties closed under stuttering?

Assumptions:
  - Multiple events can happen simultaneously
  - Intervals are closed-left, open-right, as in original system

17

Extending the Pattern System

- We have considered the following possibilities:

0. P, S -- states — Q, R -- states
1. P, S -- states — Q, R -- up edges
2. P, S -- up edges — Q, R -- states
3. P, S -- up edges — Q, R -- up edges

Note: down edges can be substituted for up edges

- We extended Absence, Existence, Universality, Precedence, and Response patterns.
- Some of properties from other patterns, e.g. Chain Precedence, are not closed under stuttering [paun,chechik'99]

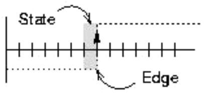
18

A Note on Edges

Definition of an edge:

$\uparrow A = \neg A \wedge \circ A$

Thus, an edge is detected in a state before it occurs.



Example: P always becomes true after Q.

Formulations:

- $\Box(Q \Rightarrow \Box P)$ if Q and P are states
- $\Box(\uparrow Q \Rightarrow \circ\Box P)$ if P is a state and Q is an event

Extension of Patterns - Existence Pattern

- P Exists Before R
  0. $\Diamond R \Rightarrow \neg(\neg P\ U R)$
  1. $\Diamond\uparrow R \Rightarrow (\neg\uparrow R\ U P)$
  2. $\Diamond R \Rightarrow \neg(\neg\uparrow P\ U R)$
  3. $\Diamond\uparrow R \Rightarrow \neg(\neg\uparrow P\ U \uparrow R)$
- P Exists Between Q and R
  0. $\Box(Q \wedge \Diamond R \Rightarrow \neg(\neg P\ U R) \wedge \neg R)$
  1. $\Box(\uparrow Q \wedge \Diamond\uparrow R \Rightarrow \circ(\neg\uparrow R\ U P) \wedge \neg\uparrow R)$
  2. $\Box(Q \wedge \Diamond R \Rightarrow \neg(\neg\uparrow P\ U R) \wedge \neg R)$
  3. $\Box(\uparrow Q \wedge \Diamond\uparrow R \Rightarrow \neg(\neg\uparrow P\ U \uparrow R) \wedge \neg\uparrow R)$

Using the Pattern System: Example

Example property:

The robot must weigh the blank after pickup from the feedbelt, but before depositing it on the press.

Variables:

(state) mgn - true when the magnet is on

(state) scl - the scale reports a successful weighing

This is the Existence pattern: weighing (state) must happen between (events) pickup and deposit. Scope is Between R and Q.

Pattern Formula:

$$\Box(\uparrow Q \wedge \Diamond\uparrow R \Rightarrow \circ(\neg\uparrow R\ U P) \wedge \neg\uparrow R)$$

Resulting Formula:

$$\Box(\uparrow mgn \wedge \Diamond\downarrow mgn \Rightarrow \circ(\neg\downarrow mgn\ U\, scl) \wedge \neg\downarrow mgn)$$

Proving Closure Under Stuttering

- Can use properties of closure under stuttering, the algebra of edges, and rules of logic to show

  $(\ll P \gg \wedge \ll Q \gg \wedge \ll R \gg) \Rightarrow$
  $\ll \Box(\uparrow Q \wedge \Diamond\uparrow R \Rightarrow \circ(\neg\uparrow R\ U P) \wedge \neg\uparrow R) \gg$

  in roughly 8 steps (see paper) completely syntactically.
- We proved all new edge-based formulas for closure under stuttering.
- Users can use these without worrying

Summary of the Problem

- The "next" operator in LTL is required for reasoning about events
- " "next" is present => the result is not closed under stuttering"
- Solution: introduce extra variables to simulate events:
  - Clutter the model, make harder to analyze
  - Results of verification cannot be interpreted correctly, without complete understanding of the modeling language and LTL. So, novice users will be making mistakes!!!

Summary of Solution

- We introduced the notion of edges for LTL
- We provided a set of theorems that enable syntax-based analysis of a large class of formulas for closure under stuttering.
- Such theorems can be added to a theorem-prover for mechanized checking.
- The language is not closed (unlike "next"-free LTL)
- But it can express properties useful in practice:
  - Properties of Production Cell [Paun,Chechik,Biechele'98]
  - Property patterns + events [Paun,Chechik'99]
- For more information:

  `http://www.cs.toronto.edu/~chechik/edges.html`