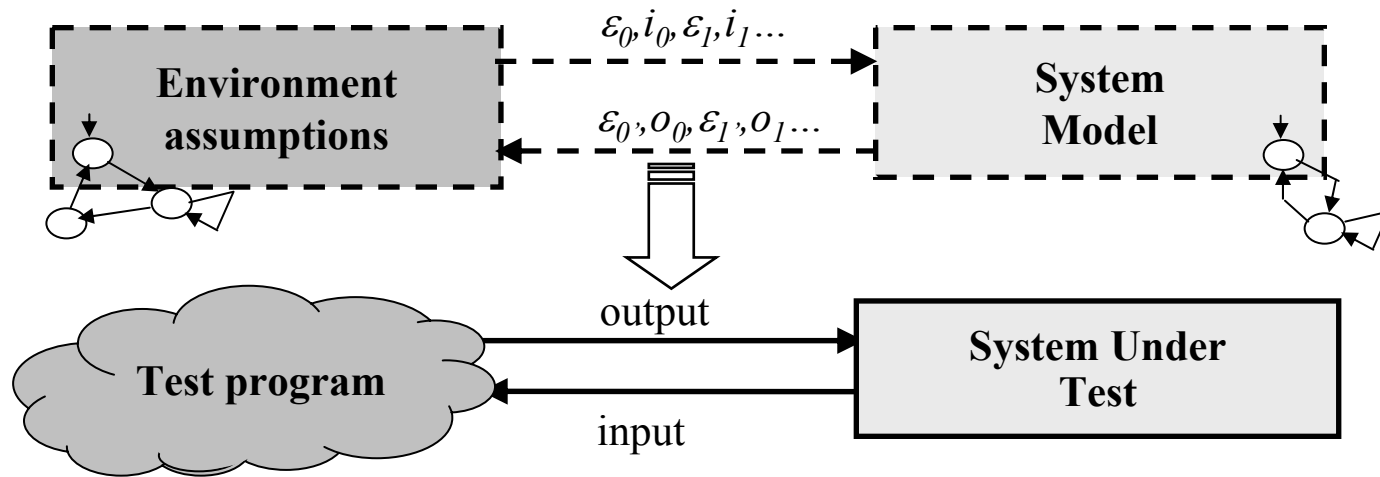


Specifying Test Cases using Observer Automata

*Johan Blom, Anders Hessel,
Bengt Jonsson, and Paul Pettersson*
Department of Information Technology
Uppsala University



Model Based Testing



- Extended finite state machine (EFSM):
 - System specification
 - Environment assumptions
- State-space exploration algorithm to generate traces
- Traces interpreted as (converted to) test cases



Coverage Criteria

- Coverage criteria of models (not code coverage)
- Systematic and automatic test selection
- Many coverage criteria have been suggested, e.g.:
 - ✱ Structural: *locations, edges, ...*
 - ✱ Data-flow: *definition-use pairs, affect-pairs, ...*
 - ✱ Semantics: *states (symbolic), equivalence classes, ...*
- Specific test-case generation algorithms
- Witness generation by model-checking



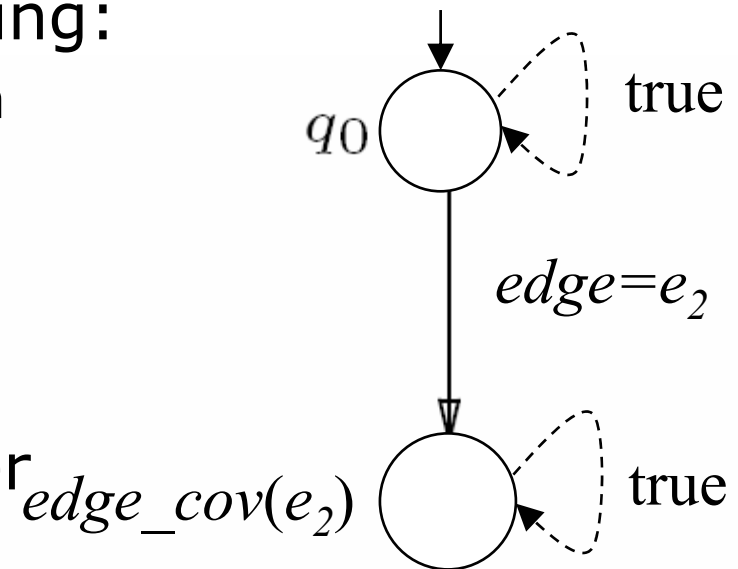
Summary of this work

- Specification language for coverage criteria:
 - ✱ Monitor and accept traces
 - ✱ Observers with parameters
 - ✱ Combine structural, data flow, and semantic CC
 - ✱ All sub-traces are considered
- Method for calculating coverage:
 - ✱ One pass exploration of state-space
 - ✱ Subset construction of observer location set
 - ✱ Alt. Monitor test execution



Observers

- Locations: q_0 , $edge_cov(e_2)$
- Edges with predicates using:
 - Info of last EFSM-transition
e.g. $edge=e_1$, $input = i_1$
 - Info of new EFSM-state
e.g. $loc=l_2$, (projections)
 - Observer parameters
- Predicates evaluated after each EFSM-transitions
- Superposed onto EFSM
 - Gives the MC a combined state

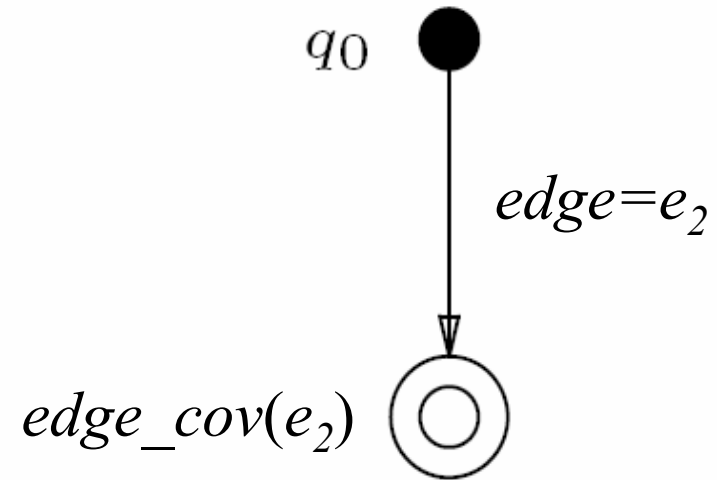


edge coverage of e_2



Observer Syntax

- ● = initial location
+ self loop
- ⊙ = accepting location
+ self loop
- ○ = other locations



edge coverage of e_2

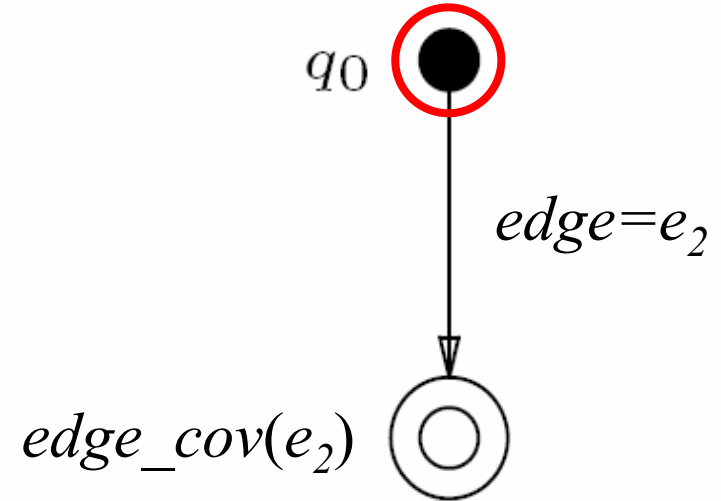
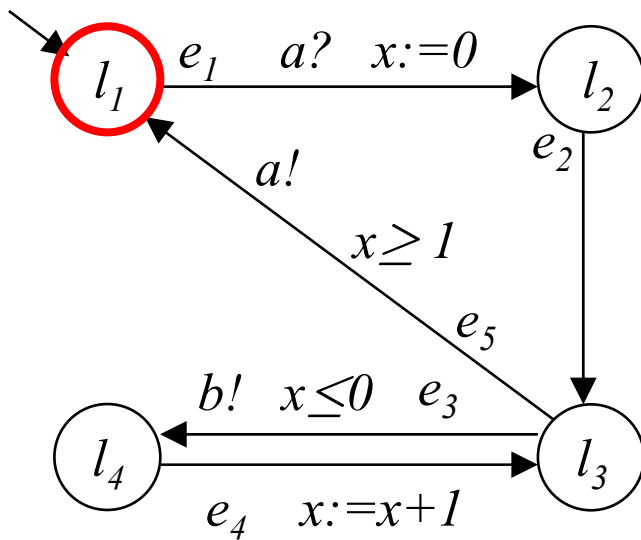


The Game

- Observer must react on every EFSM-transition
- Observer cannot stay in location unless there is a self loop with satisfied predicate
- Observer must follow every outgoing edge
- The observer state is the set of current locations
- The new observer state is calculated by subset constructions



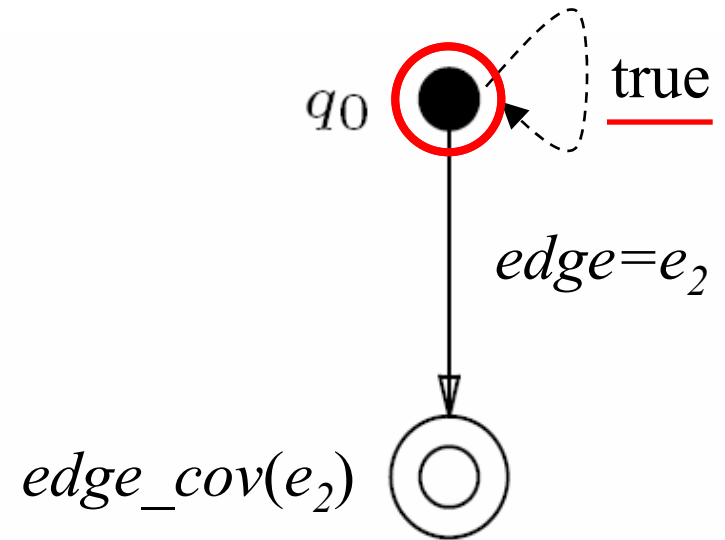
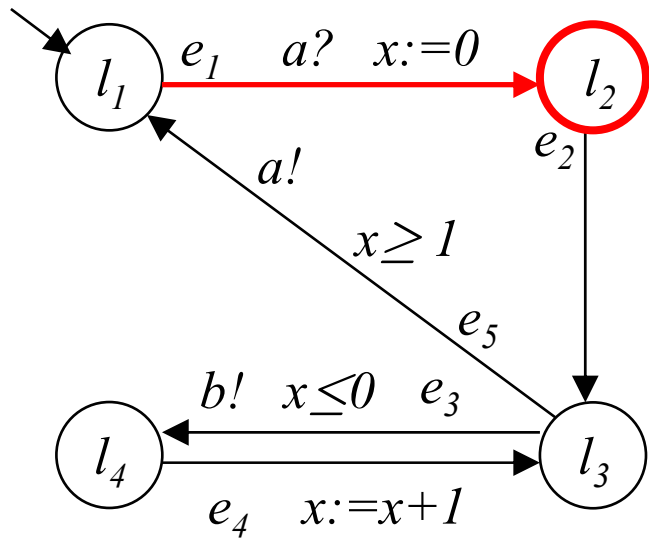
Example



edge coverage of e_2



Example

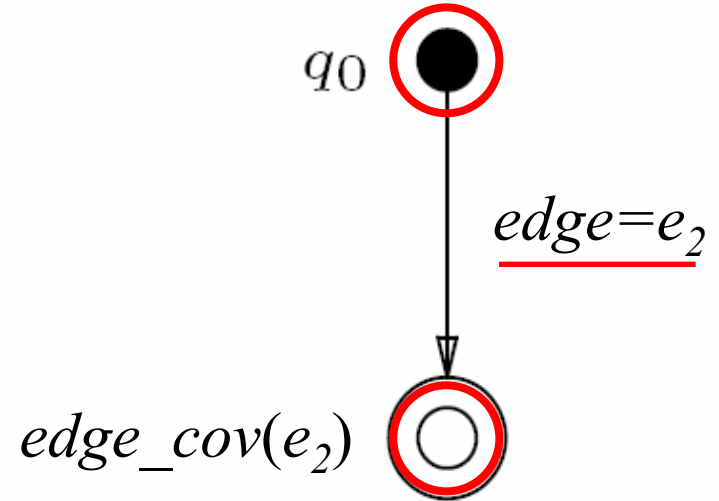
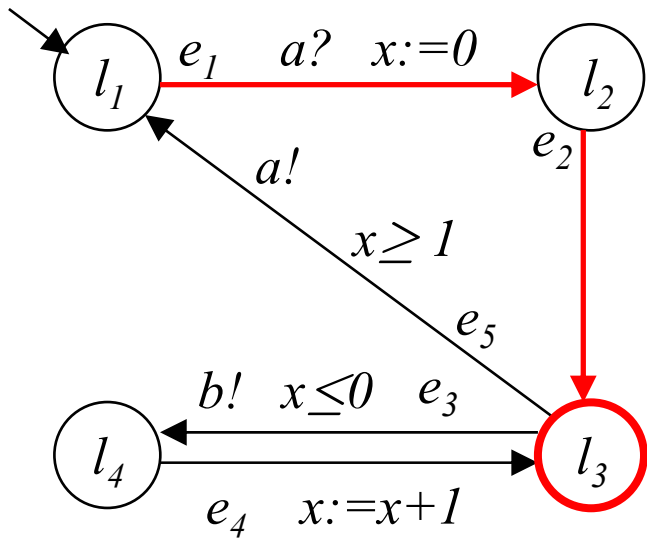


edge coverage of e_2



UPPSALA
UNIVERSITET

Example



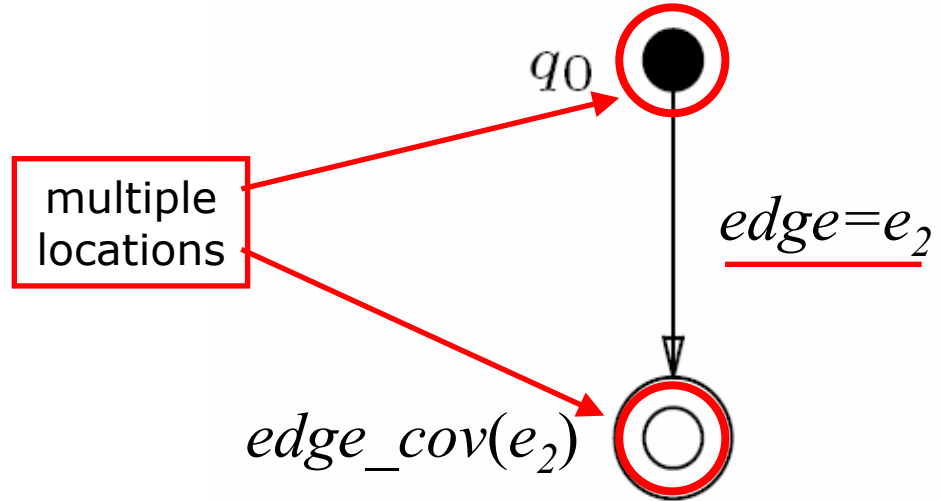
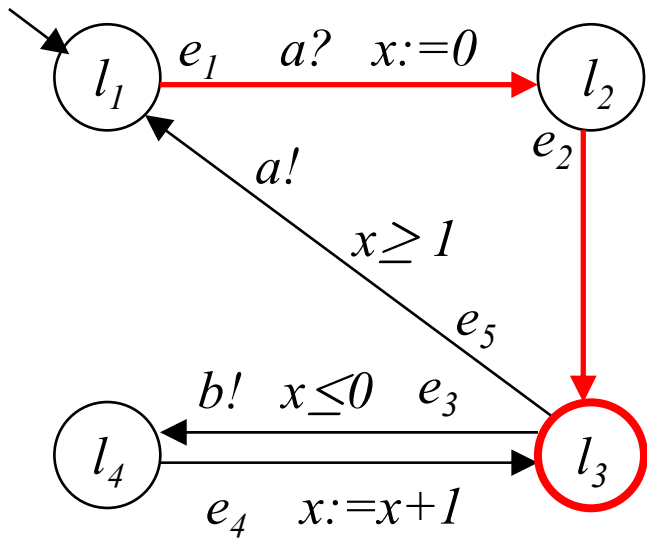
DONE!

edge coverage of e_2

Uppsala 2004



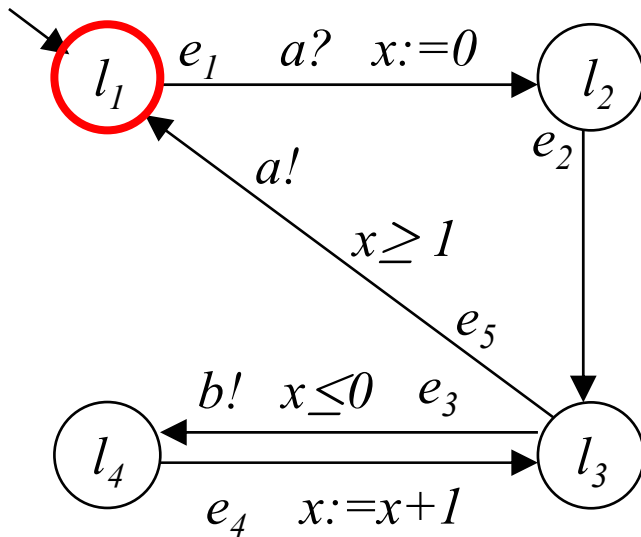
Example



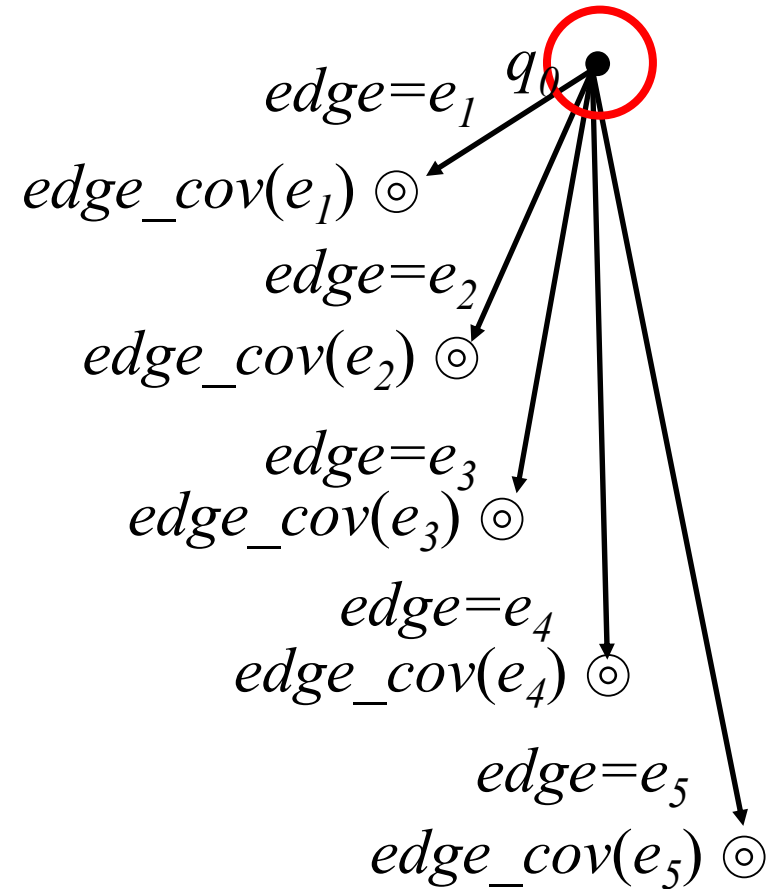
edge coverage of e_2



Coverage Criteria all-edges

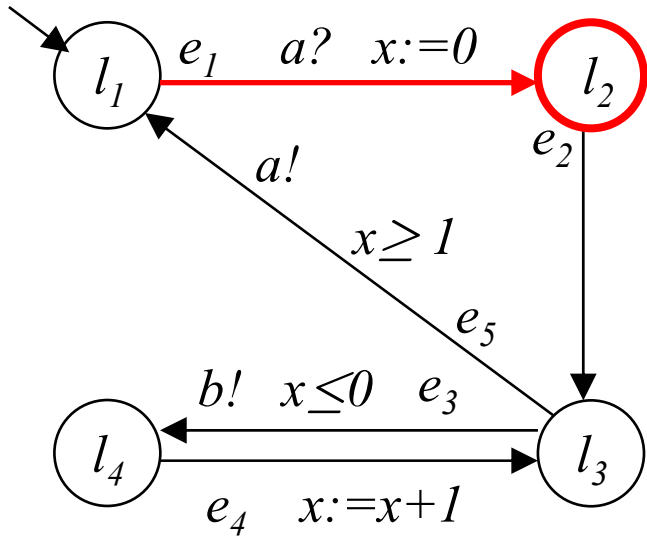


Locations: q_0

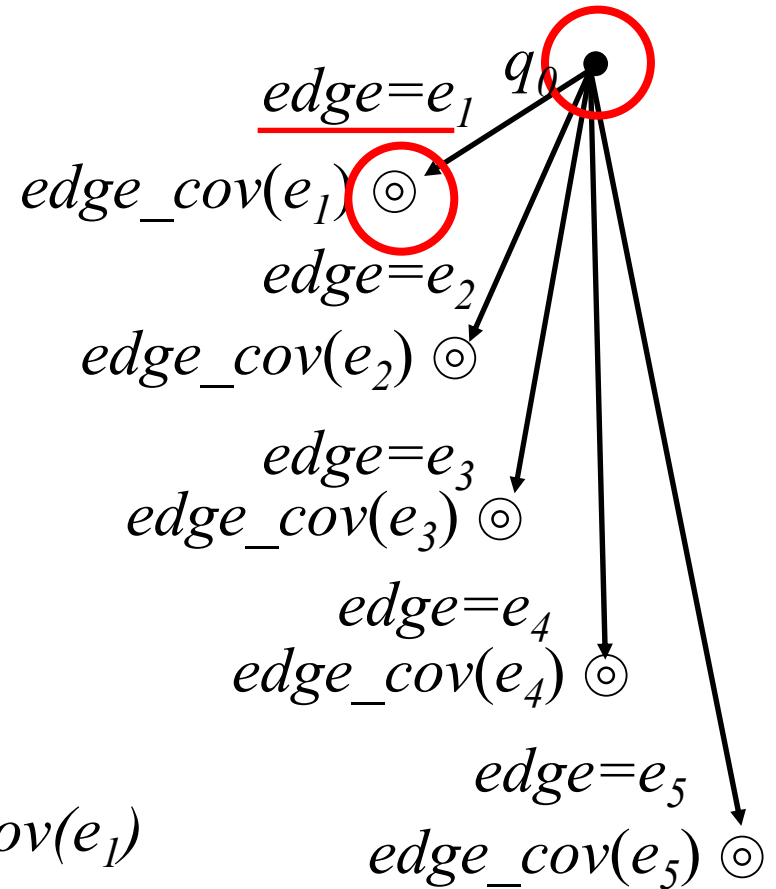




Coverage Criteria all-edges

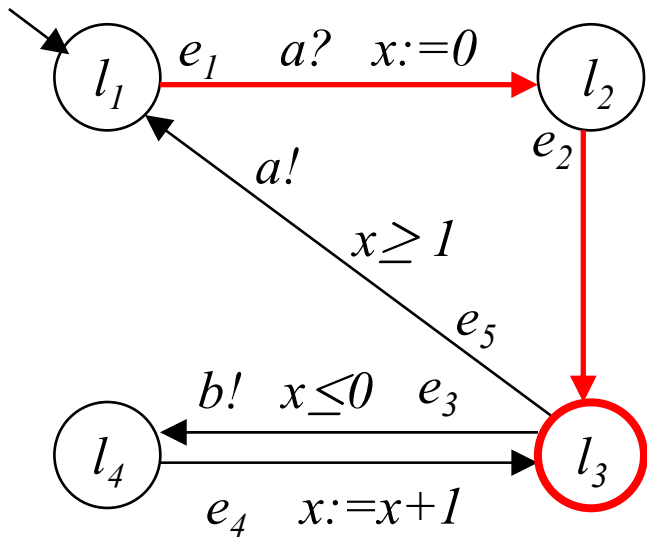


Locations: $q_0, edge_cov(e_1)$

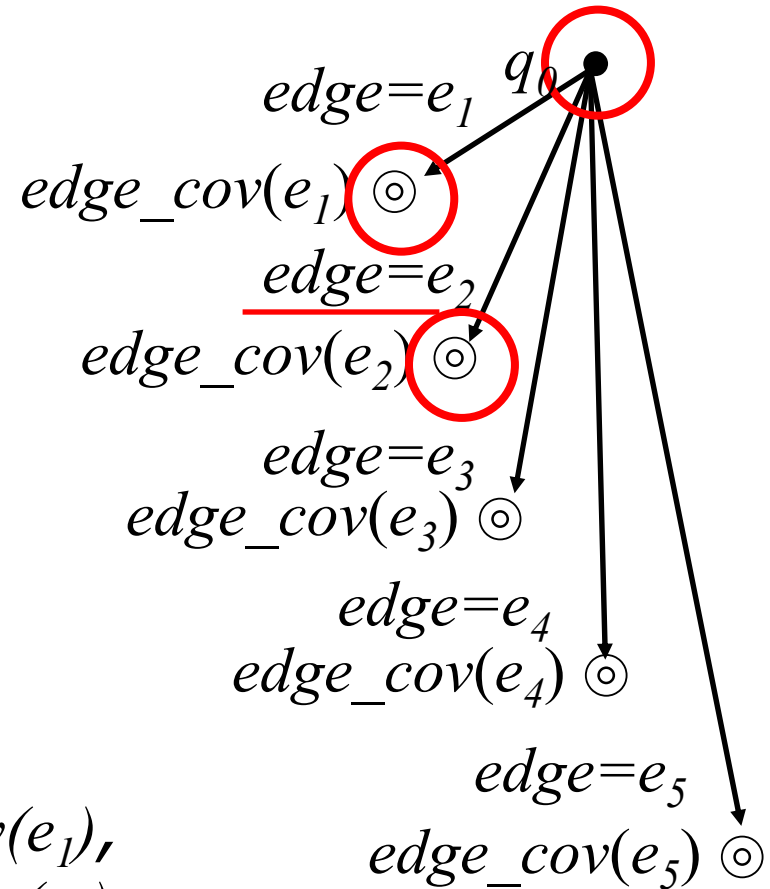




Coverage Criteria all-edges



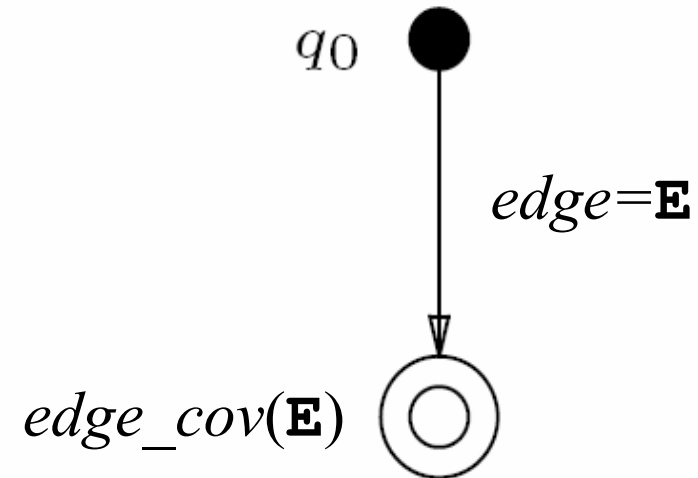
Locations: $q_0, edge_cov(e_1), edge_cov(e_2)$
...





Observers with parameters

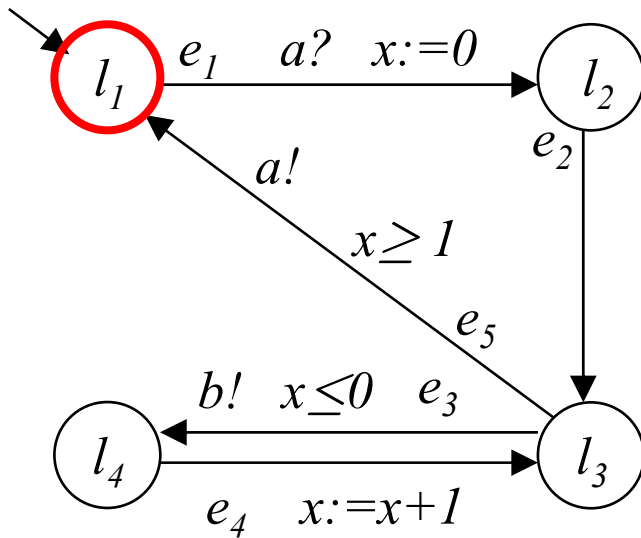
- Parameters of finite domain
- Locations with parameters:
 $q_0, edge_cov(\mathbf{E})$
- Edges with parameters
- Represents all possible instantiations of the parameters
- Syntax...
- Intuition: as many observers as edges in \mathbf{E}



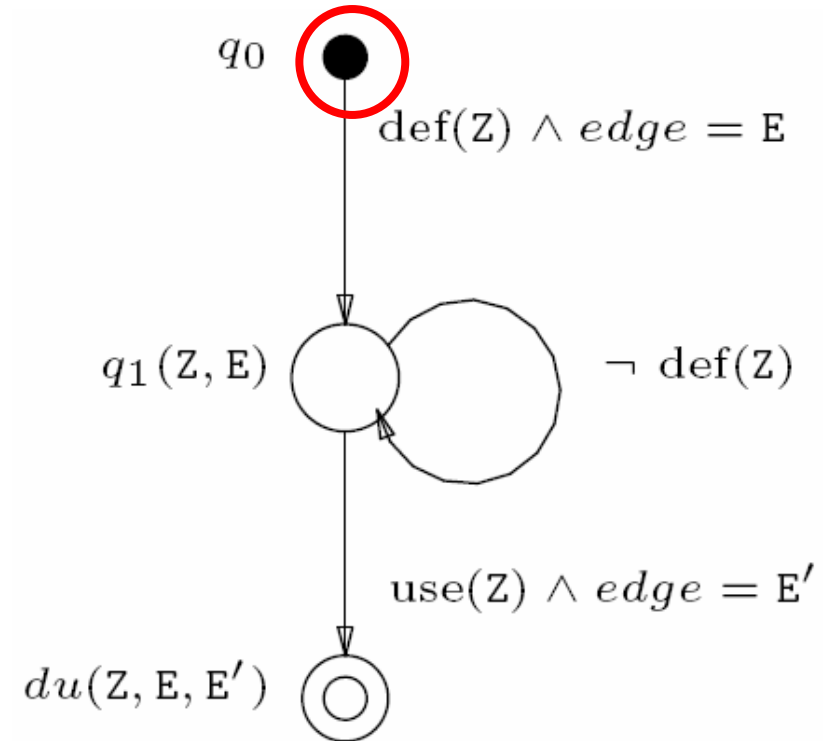
edge coverage criteria:
coverage of any edge



Definition Use pair

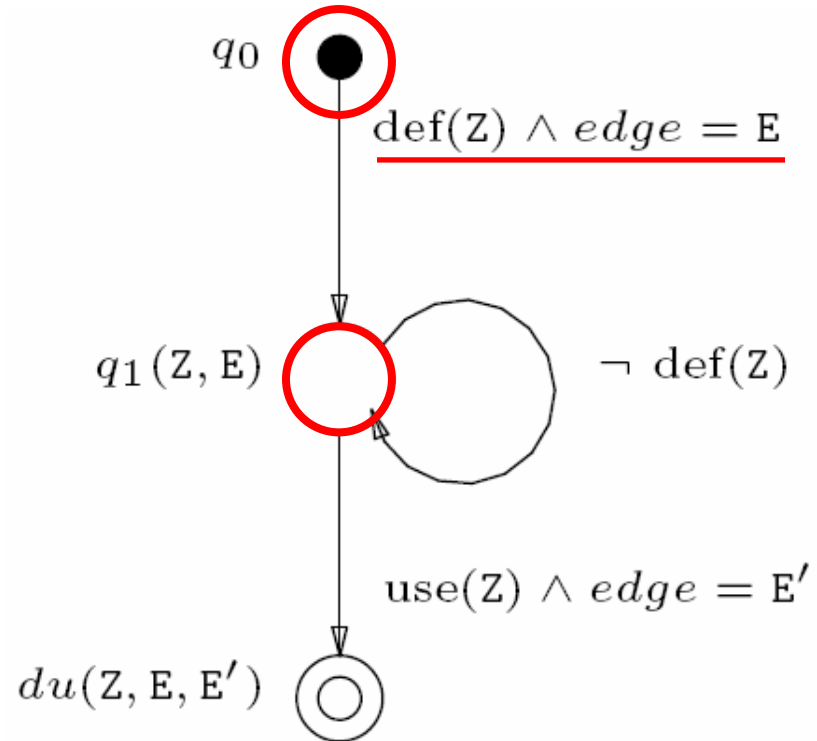
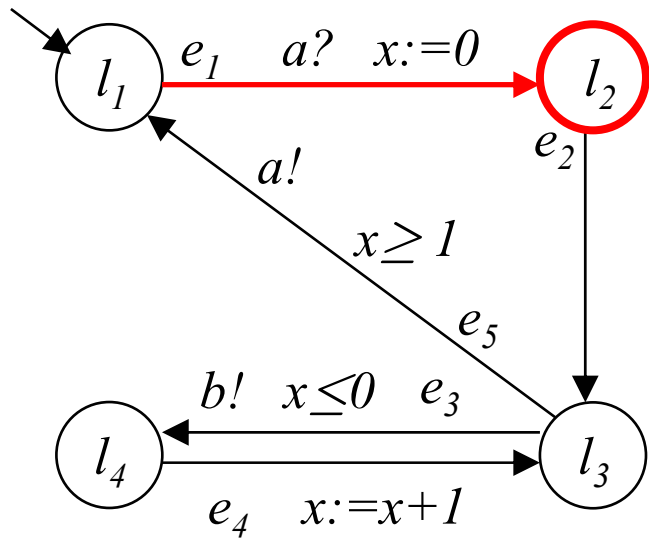


Locations: q_0





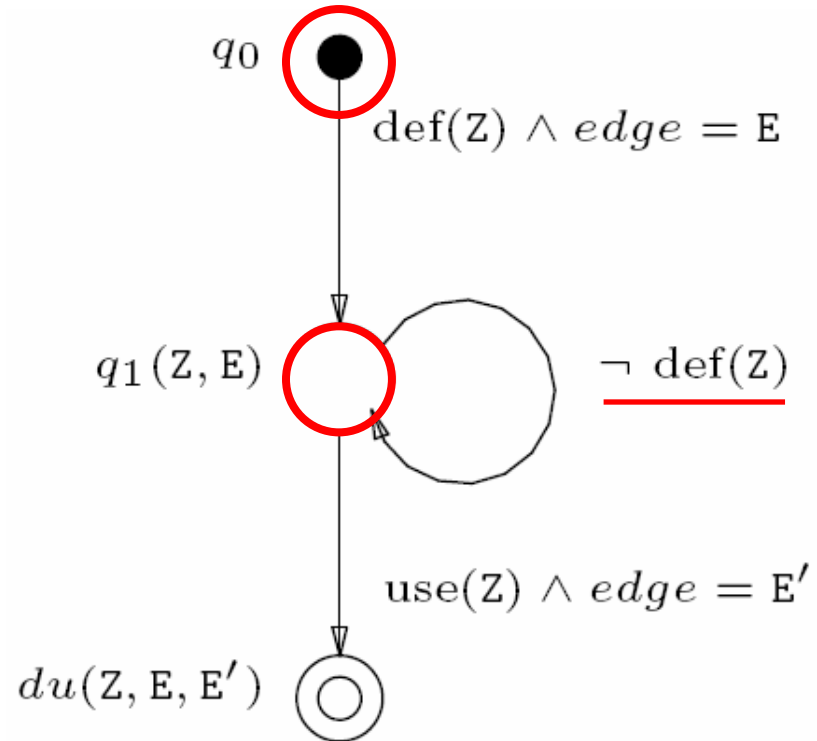
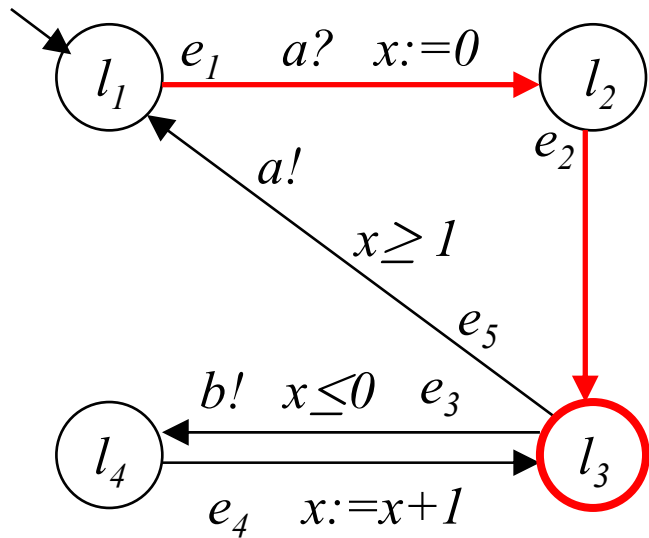
Definition Use pair



Locations: $q_0 \rightarrow q_0, q_1(x, e_1)$



Definition Use pair



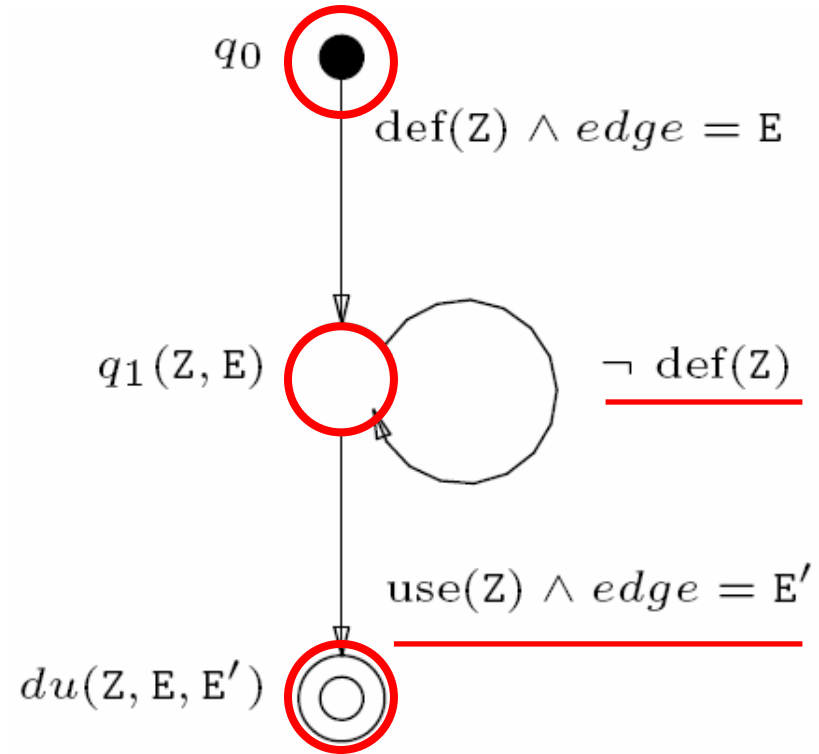
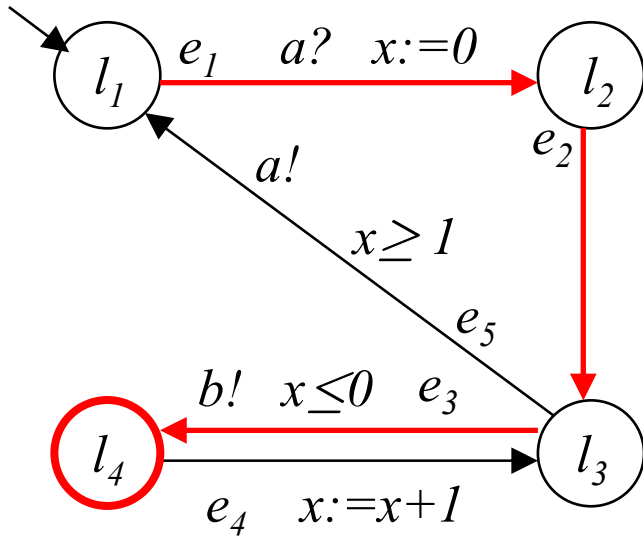
Locations:

$$q_0 \rightarrow q_0$$

$$q_1(x, e_1) \rightarrow q_1(x, e_1)$$



Definition Use pair



Locations:

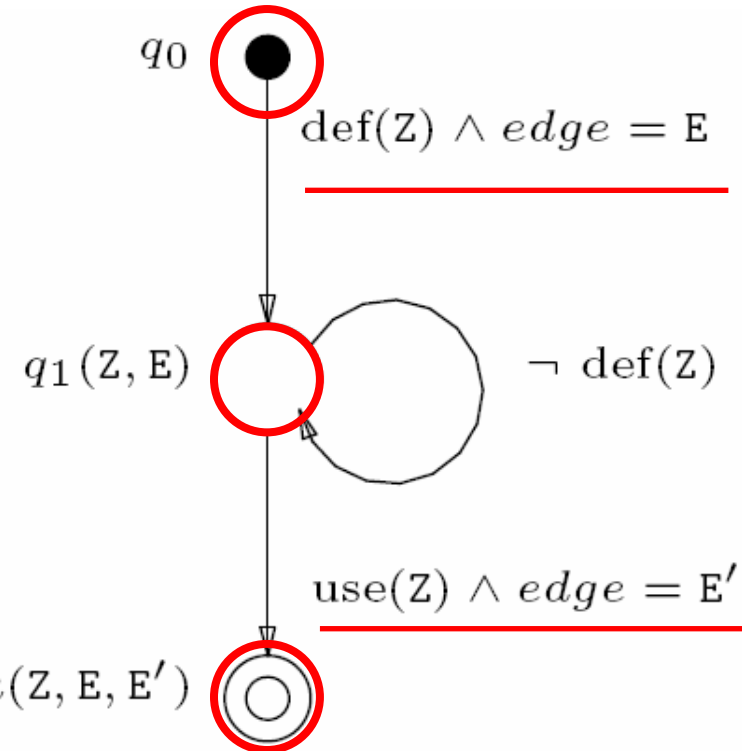
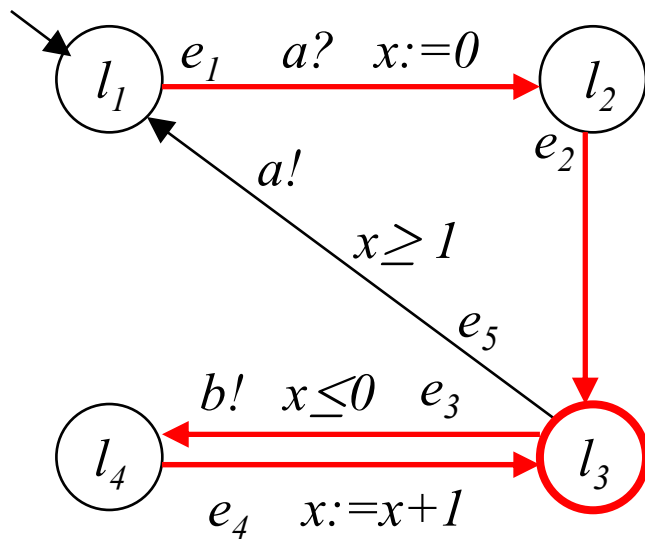
$$q_0 \rightarrow q_0$$

$$q_1(x, e_1) \rightarrow q_1(x, e_1), du(x, e_1, e_3)$$



Definition Use pair

No sequential update,
use before def



Locations:

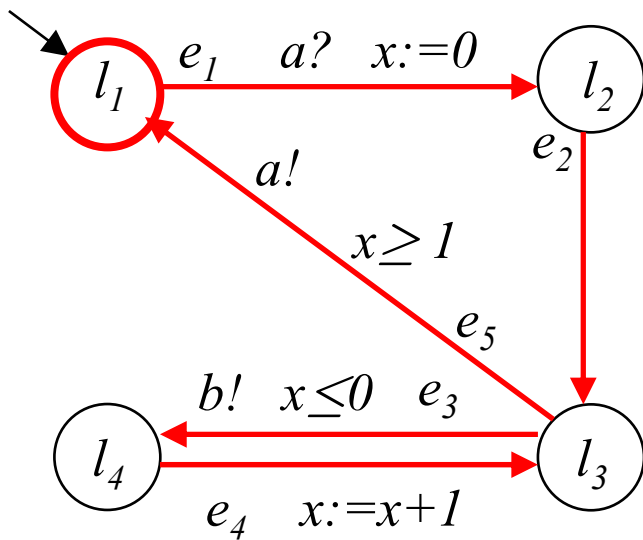
$$q_0 \rightarrow q_0, q_1(x, e_4)$$

$$q_1(x, e_1) \rightarrow du(x, e_1, e_4)$$

$$du(x, e_1, e_3) \rightarrow du(x, e_1, e_3)$$



Definition Use pair



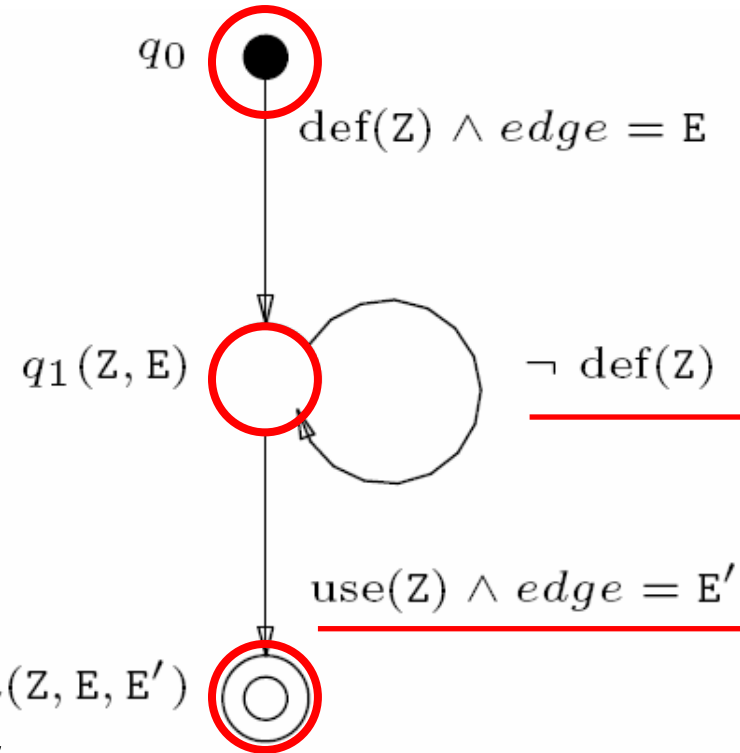
Locations:

$$q_0 \rightarrow q_0$$

$$q_1(x, e_4) \rightarrow q_1(x, e_4), du(x, e_4, e_5)$$

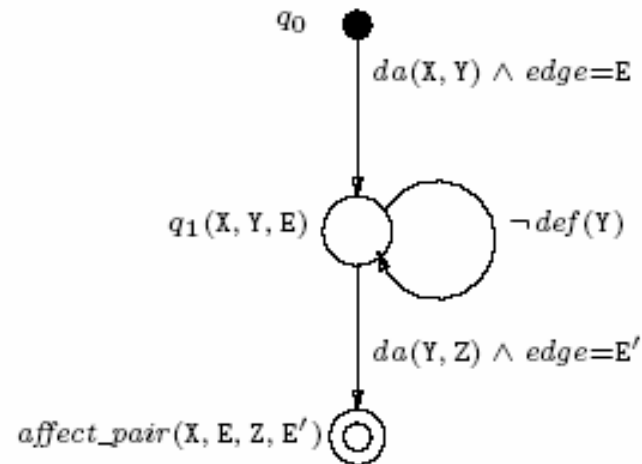
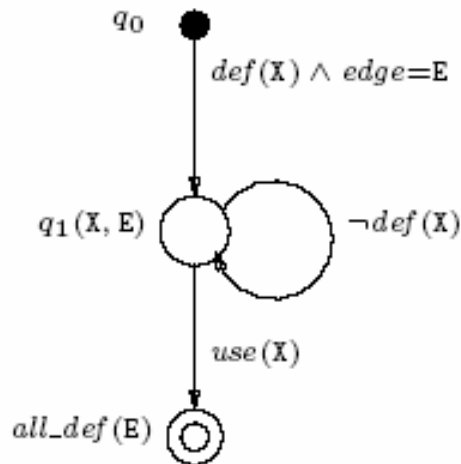
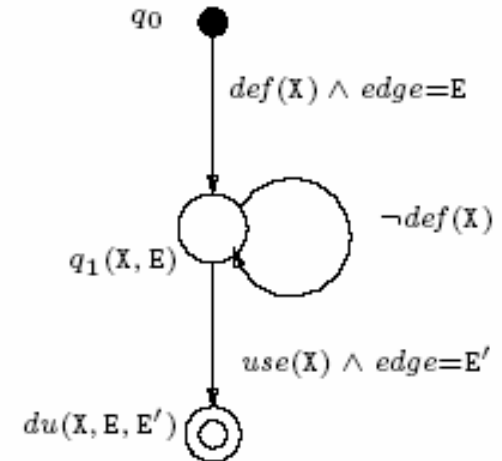
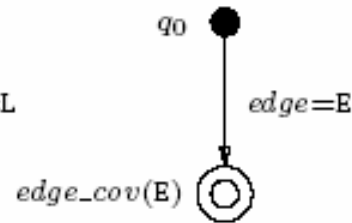
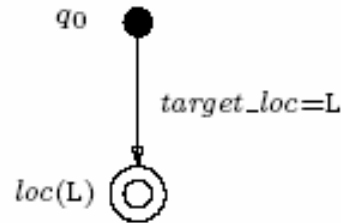
$$du(x, e_1, e_4) \rightarrow du(x, e_1, e_4)$$

$$du(x, e_1, e_3) \rightarrow du(x, e_1, e_3)$$





Example observers





MC with Observers

- State of superposition: $\langle \langle l, \sigma \rangle \parallel Q \rangle$

EFSM state Superposition Observer locations
- Idea: find trace
 - $\langle \langle l_0, \sigma_0 \rangle \parallel \{q_0\} \rangle \rightsquigarrow \dots \rightsquigarrow \langle \langle l, \sigma \rangle \parallel Q \rangle$
 - such that Q includes required number of accepting locations
- State-space exploration algorithm
- Optimization algorithm: max coverage
 - Gets all feasible accepting locations (if reset permitted)
 - Terminates: finite state $|EFSM_{states}| * 2^{|Q|}$
 - $|Q_{acc}|$ monotonically increasing (self loop)



Pruning when optimizing

- If $\langle l, \sigma \rangle = \langle l', \sigma' \rangle$ and $Q \setminus Q_{\text{acc}} = Q' \setminus Q'_{\text{acc}}$ then if $Q_{\text{acc}} \subseteq Q'_{\text{acc}}$ we can safely ignore the successors of $\langle \langle l, \sigma \rangle \parallel Q \rangle$
- Same is true if $\langle l, \sigma \rangle \subseteq \langle l', \sigma' \rangle$ for symbolic states



Subset construction (bitvector)

- Q bitvector,
where $Q[i]=\text{true}$ iff q_i in the location set
- Let q' be one bit after update
- Idea:
 - $q'=1$ if one ingoing edge can be taken
 - $q'=0$ otherwise

- Bit-wise:

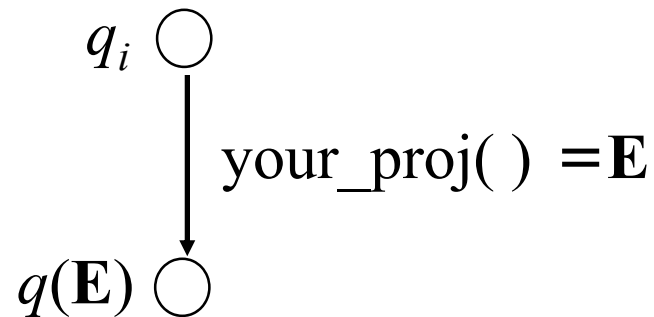
$$f_{q'}(q') = \bigvee_{\langle b, q \rangle \in \text{in}(q')} q \wedge b$$

$$\text{in}(q') = \{ \langle b, q \rangle \mid q \xrightarrow{b} q' \in B \}$$



Projection parameters

- Projections from the EFSM-state can be calculated from the EFSM variables and location(s)
- E.g. $\text{val}(\text{"x"}) \% 5 = E$ etc.





Conclusion and Future

- Observers as specification language of coverage criteria
- Complex criteria can be written in an easy way
- Different types of coverage criteria can be combined
- Implement in UPPAAL a version extended for test-case generation using observers
- Apply in case study with Ericsson: WAP gateway
- Extend observers with notion of time (observers may split DBMs)



UPPSALA
UNIVERSITET

Information Technology



Uppsala 2004

Nordic Workshop
on Programming Theory

END

- Thanks for you attention