Verification of Requirement Specification Using Counter Automata

K Vasanta Lakshmi and K V Raghavan
Indian Institute of Science, Bangalore

Outline

Outline

- Motivation

Outline

- Motivation
- Our approach

Outline

- Motivation
- Our approach
  - We give an algorithm which is similar to symbolic model checking.

Outline

- Motivation
- Our approach
  - We give an algorithm which is similar to symbolic model checking.
  - But in addition to reachability it can also answer a class of temporal properties.

Outline

- Motivation
- Our approach
  - We give an algorithm which is similar to symbolic model checking.
  - But in addition to reachability it can also answer a class of temporal properties.
- How it relates to existing work

Outline

- Motivation
- Our approach
  - We give an algorithm which is similar to symbolic model checking.
  - But in addition to reachability it can also answer a class of temporal properties.
- How it relates to existing work
  - We believe that we can define a new class of counter automata for which reachability is decidable and

Outline

- Motivation
- Our approach
  - We give an algorithm which is similar to symbolic model checking.
  - But in addition to reachability it can also answer a class of temporal properties.
- How it relates to existing work
  - We believe that we can define a new class of counter automata for which reachability is decidable and
  - This class is not subsumed by any already known such class.

Motivation

Motivation

- Our motivation for this work is a set of examples from:
  - Finance domain (Savings bank account and credit card account)
  - Filters in streaming applications
  - Simple programs (without dynamic allocation).

Motivation

- Our motivation for this work is a set of examples from:
  - Finance domain (Savings bank account and credit card account)
  - Filters in streaming applications
  - Simple programs (without dynamic allocation).
- These programs are infinite-state systems and can be naturally modeled using our restricted *counter automata*.

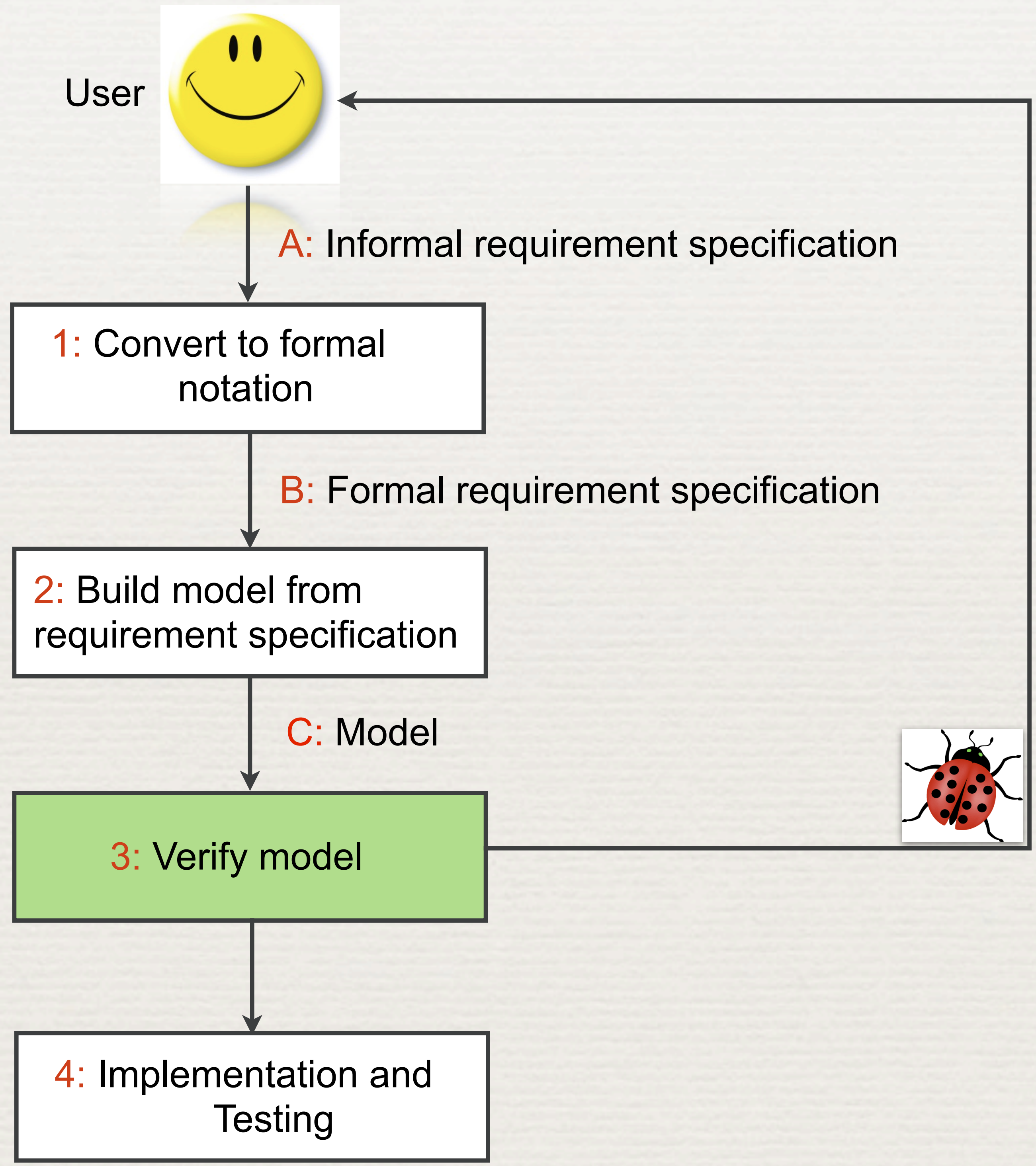
Motivation

- Our motivation for this work is a set of examples from:
  - Finance domain (Savings bank account and credit card account)
  - Filters in streaming applications
  - Simple programs (without dynamic allocation).
- These programs are infinite-state systems and can be naturally modeled using our restricted *counter automata*.
- Goal: Verification of infinite state systems that can be modeled using counter automata.

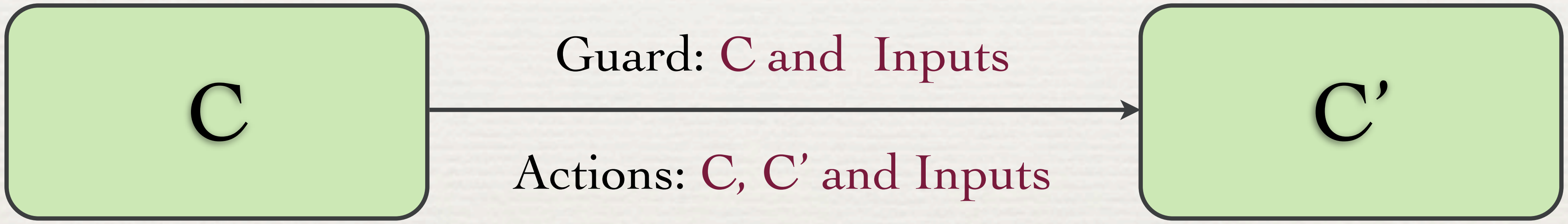
Motivation

- Our motivation for this work is a set of examples from:
  - Finance domain (Savings bank account and credit card account)
  - Filters in streaming applications
  - Simple programs (without dynamic allocation).
- These programs are infinite-state systems and can be naturally modeled using our restricted *counter automata*.
- Goal: Verification of infinite state systems that can be modeled using counter automata.
- Challenge: In infinite-state systems in general even simple property like reachability is undecidable.

Model Based Software Development Cycle

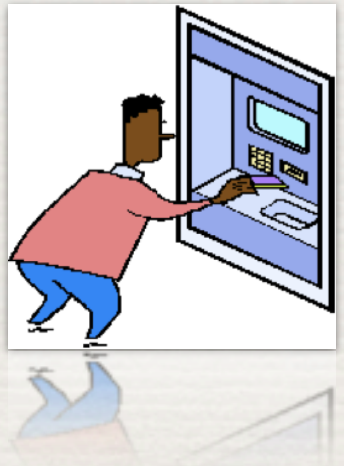


Counter Automata



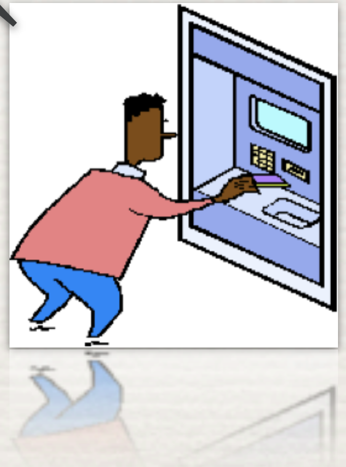
We assume Presburger arithmetic for *guards* and *actions*.

Banking Example: Requirement Specification



Banking Example: Requirement Specification

Accept deposits of any ***Amount***



Banking Example: Requirement Specification

Accept deposits of any ***Amount***

Reject withdrawals when ***Amount > 20K***

Banking Example: Requirement Specification

Accept deposits of any ***Amount***

Reject withdrawals when ***Amount > 20K***

Overdrafts are allowed

Banking Example: Requirement Specification

Accept deposits of any ***Amount***

Reject withdrawals when ***Amount > 20K***

Overdrafts are allowed

Overdrafts rejected if **Balance < -40k**

Banking Example: Requirement Specification

Accept deposits of any ***Amount***

Reject withdrawals when ***Amount > 20K***

Overdrafts are allowed

Overdrafts rejected if **Balance < -40k**

Overdrafts rejected if previous three transactions were overdrafts

Banking Example: Requirement Specification

Accept deposits of any ***Amount***

Reject withdrawals when ***Amount > 20K***

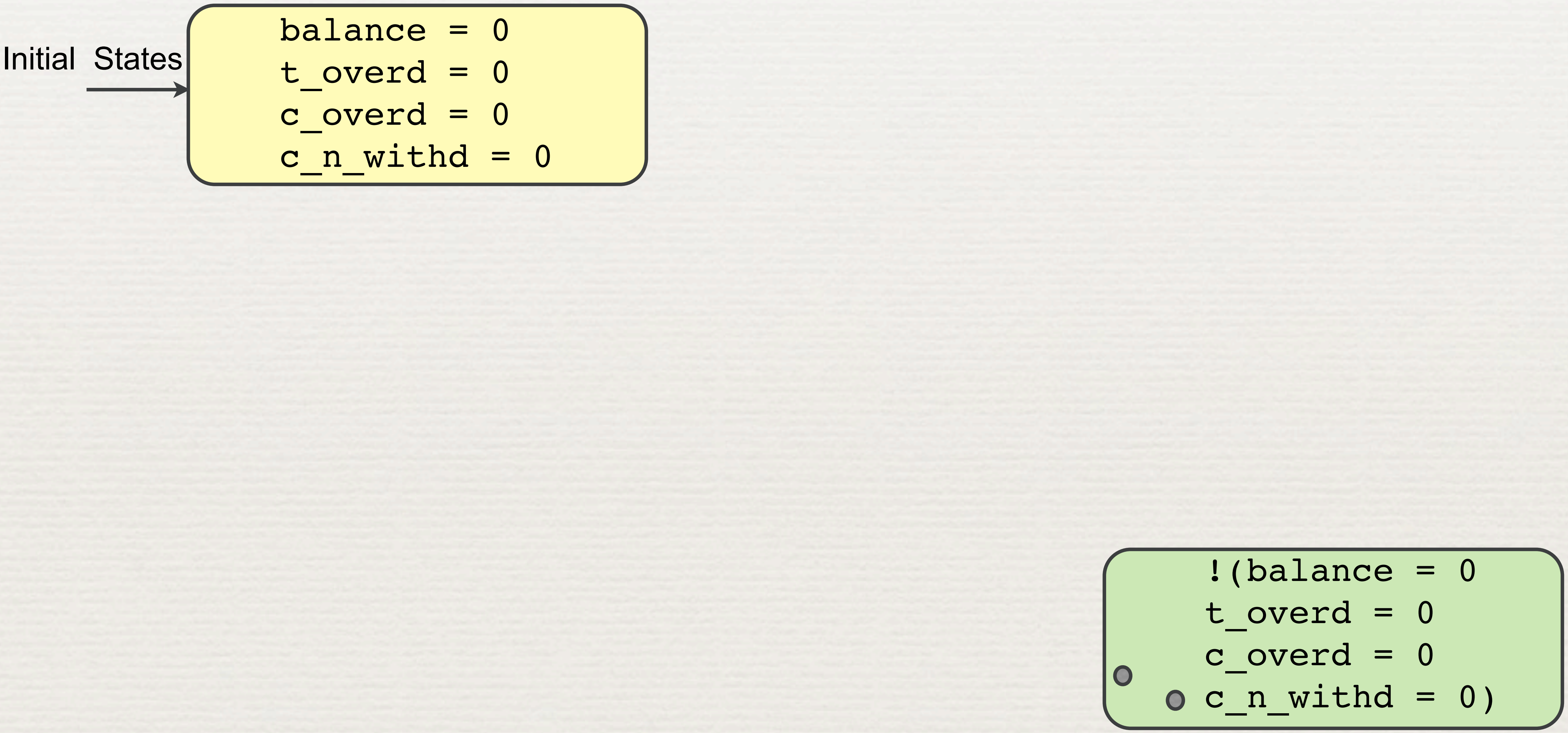
Overdrafts are allowed

After six overdrafts, no more overdrafts allowed until 12 normal withdrawals

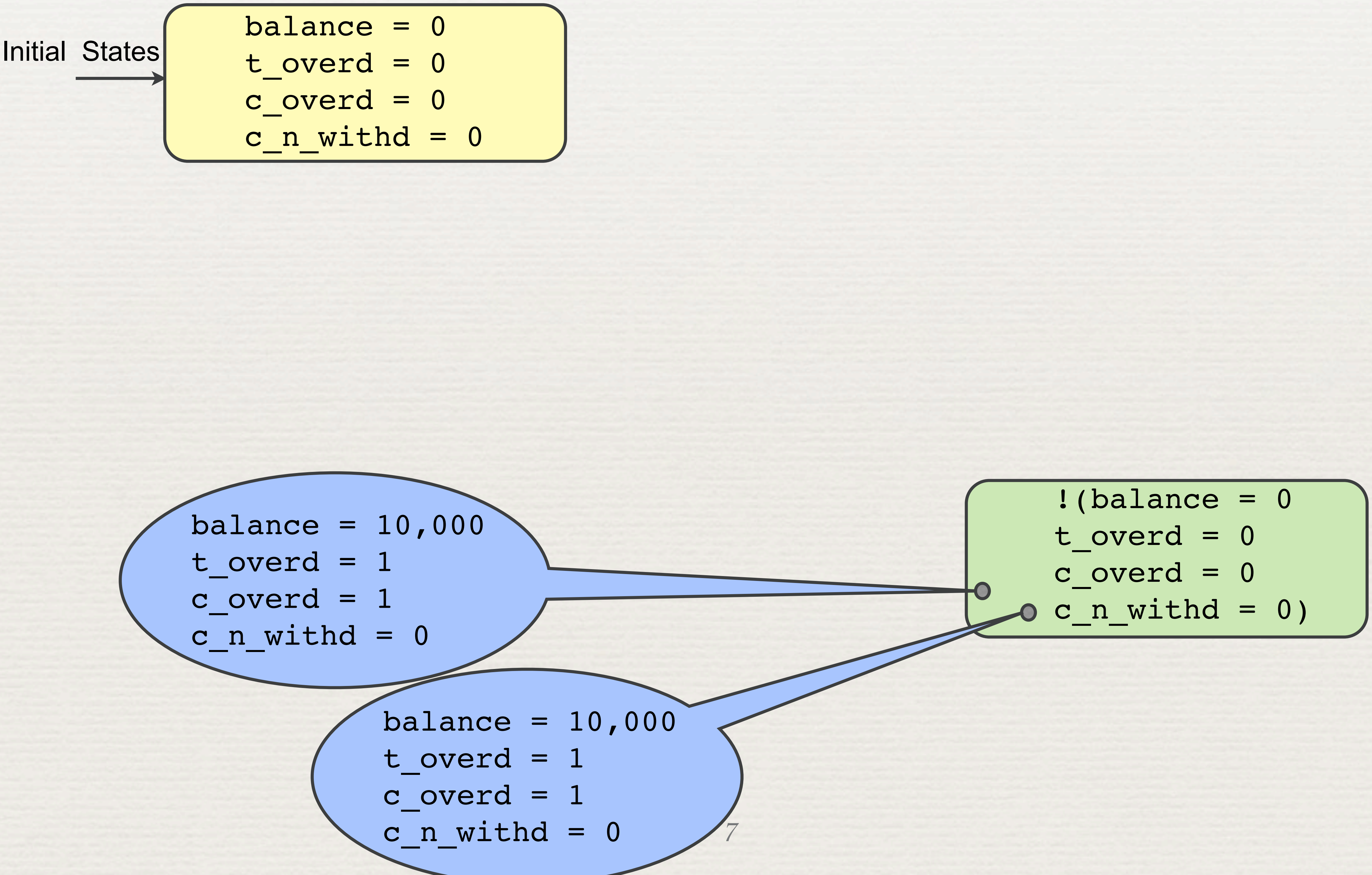
Overdrafts rejected if **Balance < -40k**

Overdrafts rejected if previous three transactions were overdrafts

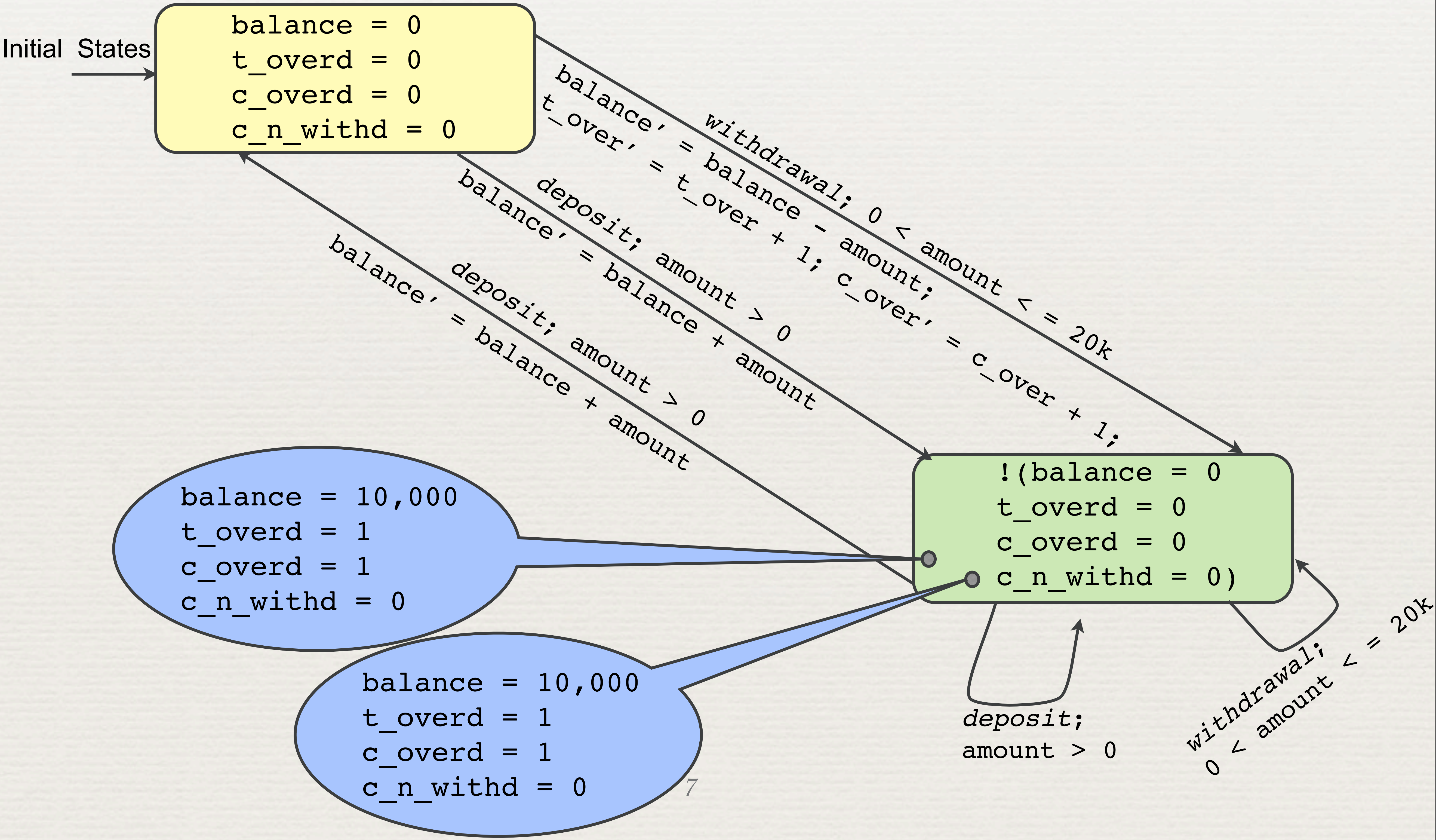
Initial Model: Counter Automata



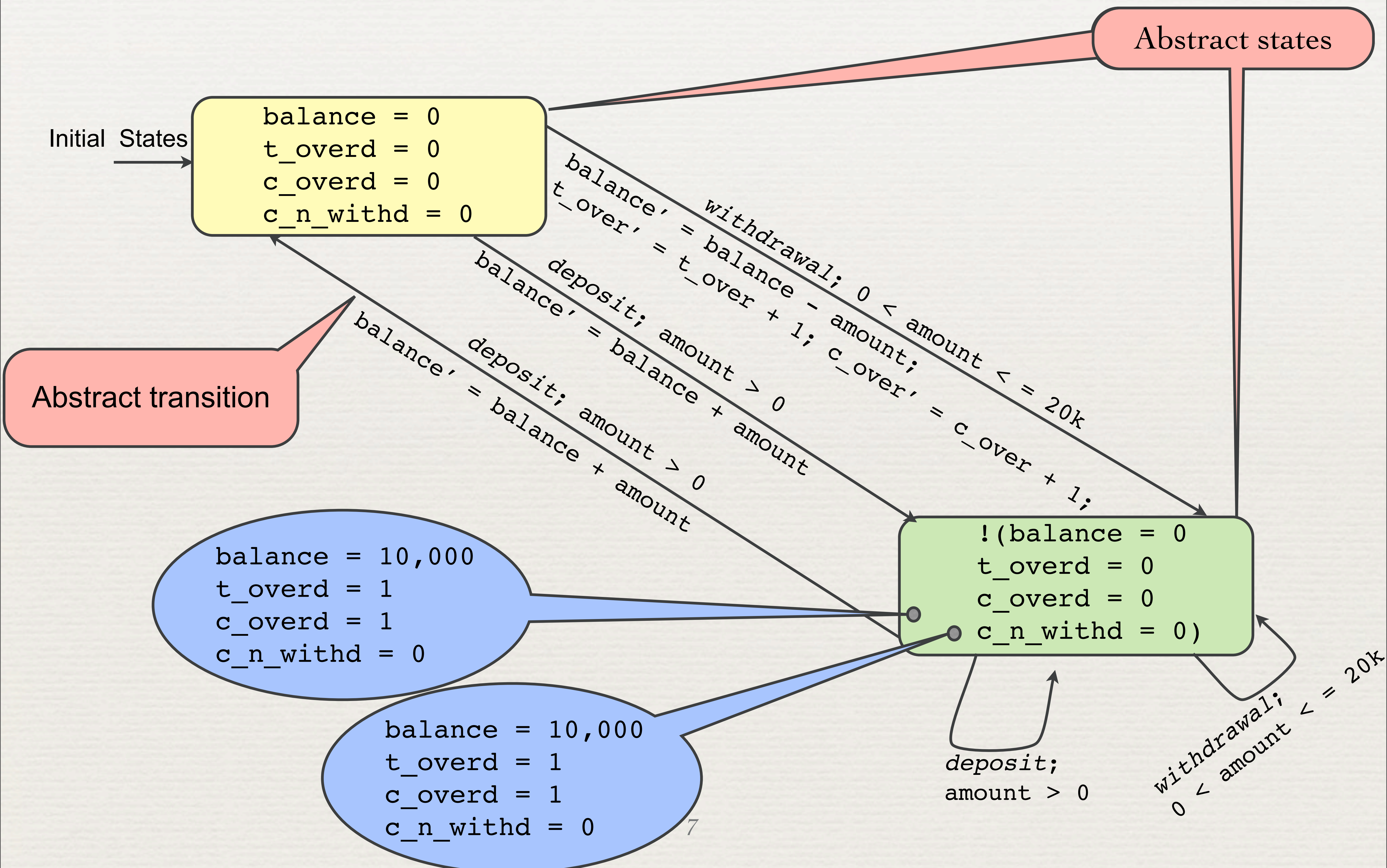
Initial Model: Counter Automata



Initial Model: Counter Automata

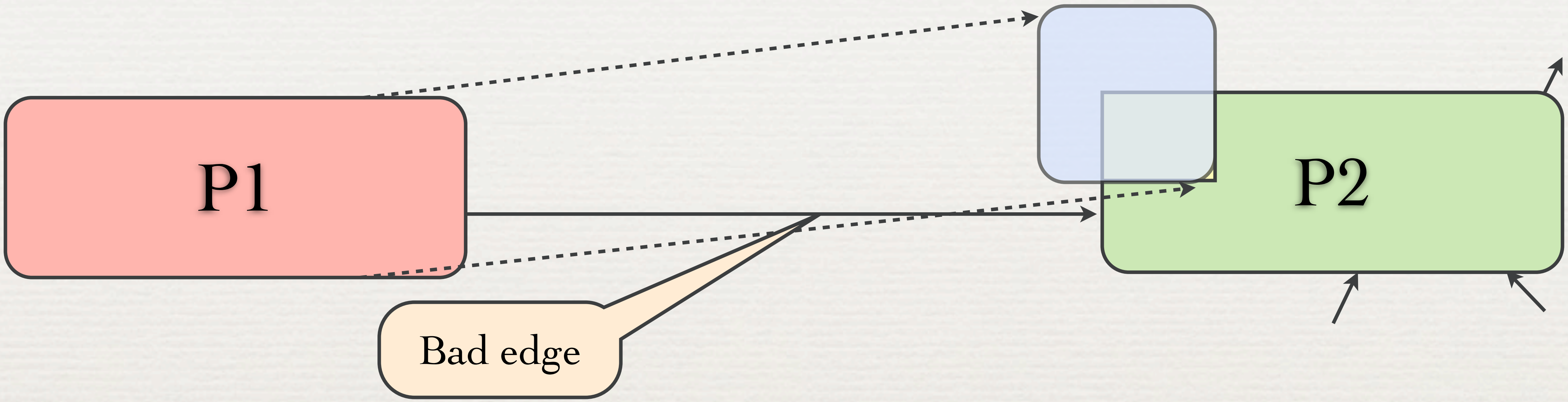


Initial Model: Counter Automata

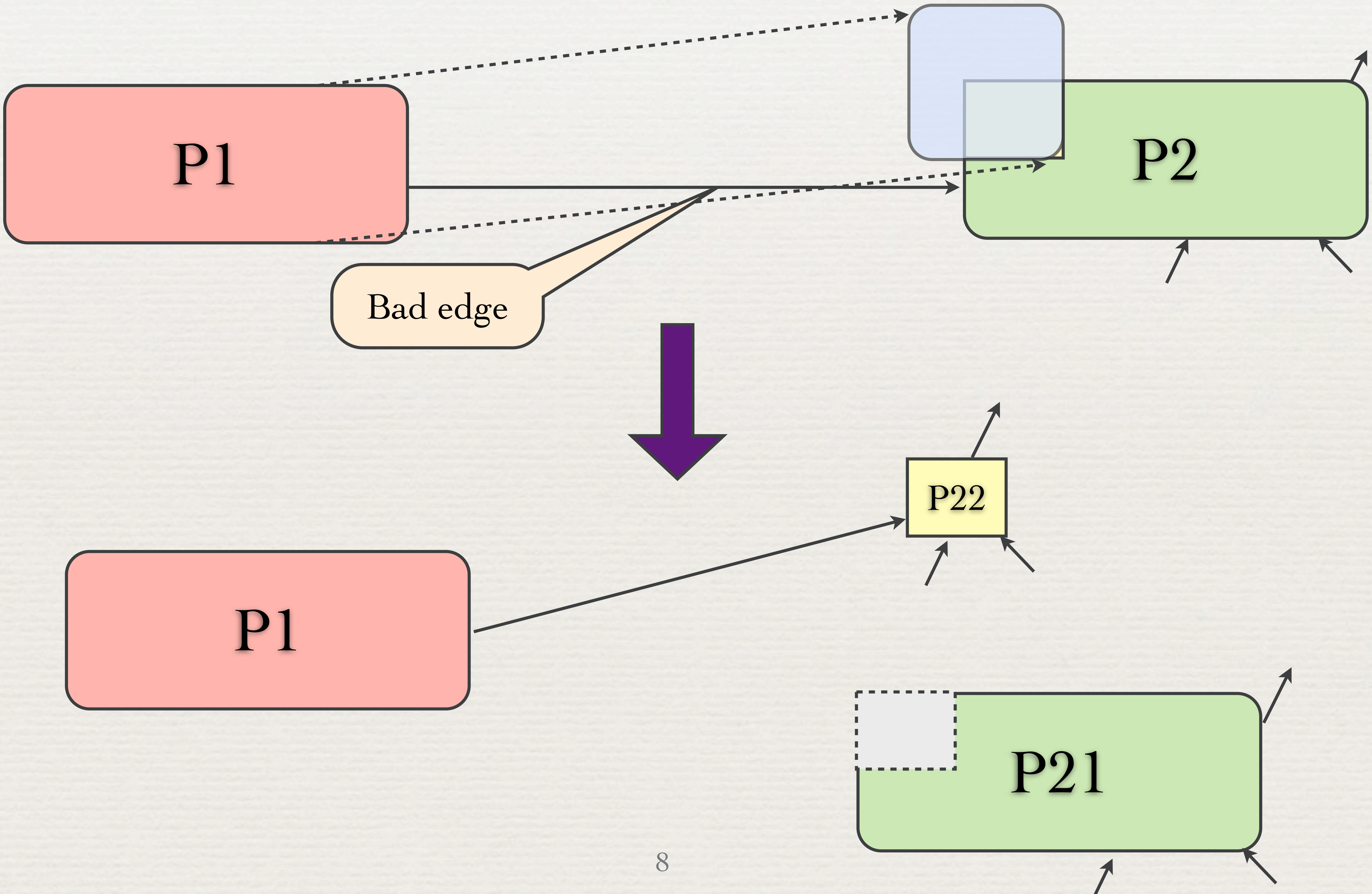


Our Approach: Partitioning Algorithm (Forward Analysis)

Our Approach: Partitioning Algorithm (Forward Analysis)



Our Approach: Partitioning Algorithm (Forward Analysis)



Our Approach: Partitioning Algorithm (Forward Analysis)

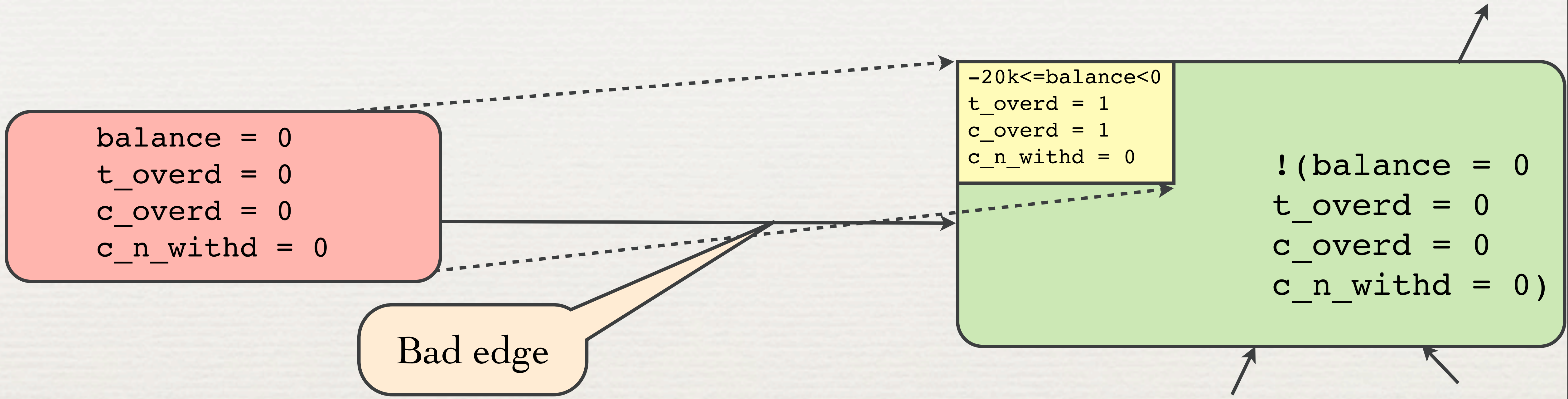
- Repeat until there are no more bad edges remaining. If algorithm terminates then we have a *finite final partitioning*.

If there is an edge between partitions P1 and P2, then every concrete state in P2 has a pre-image in P1.

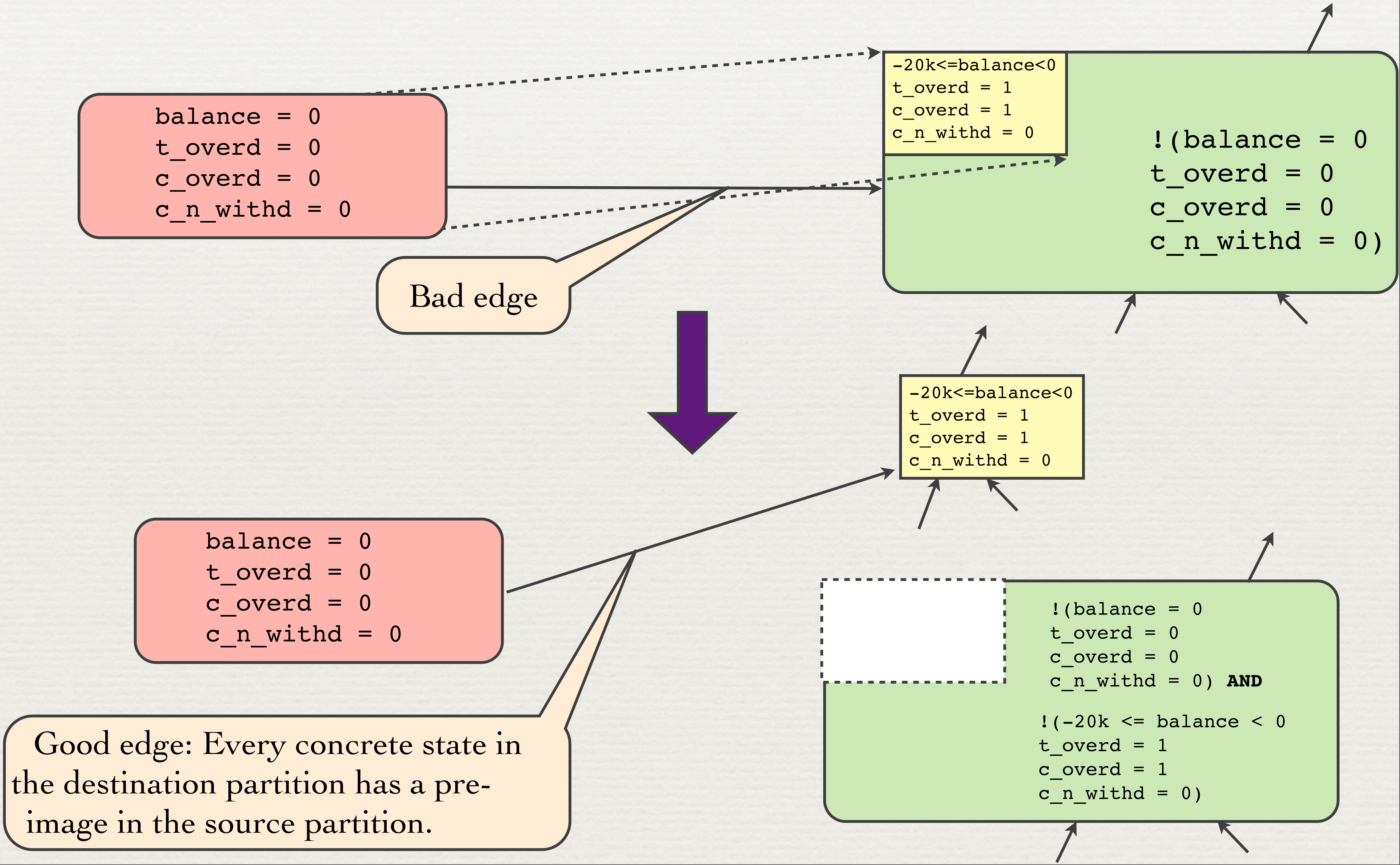
- Final partitioning is a refinement of partitioning created by symbolic model checking.

- Terminates on a subclass of systems on which symbolic model checking terminates.

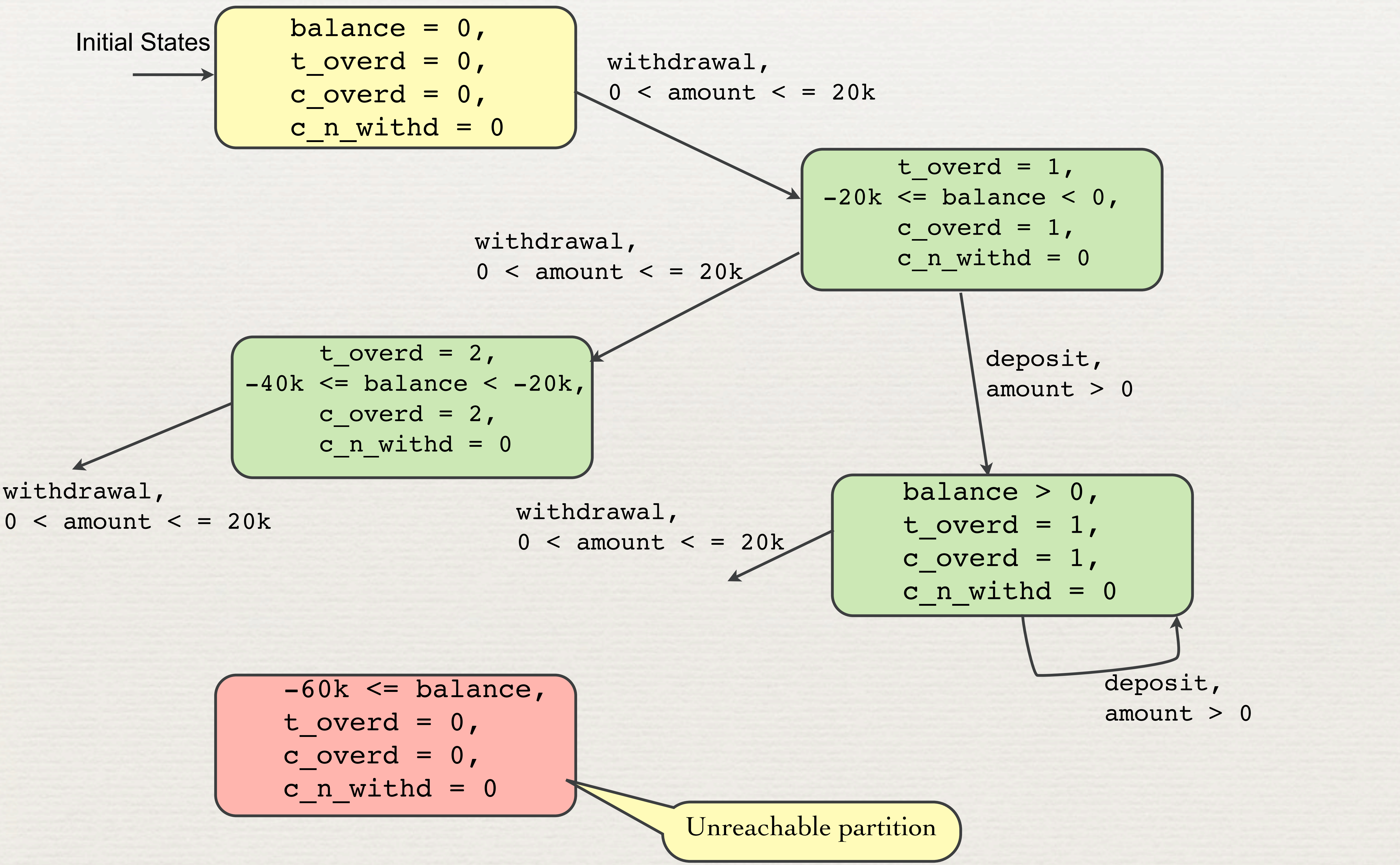
Applying our algorithm on Banking Example



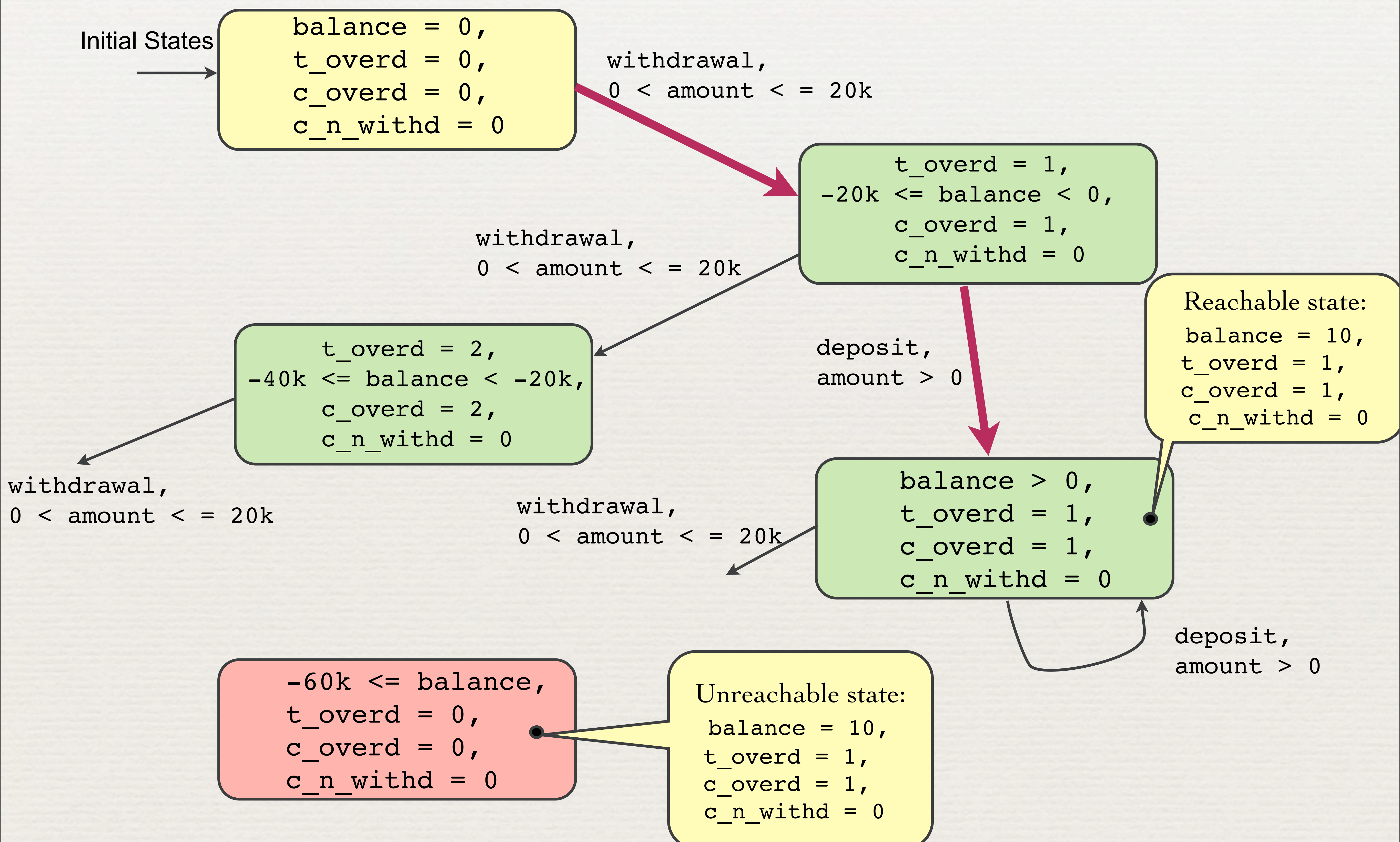
Applying our algorithm on Banking Example



Applying our algorithm on Banking Example: Part of final partitioning

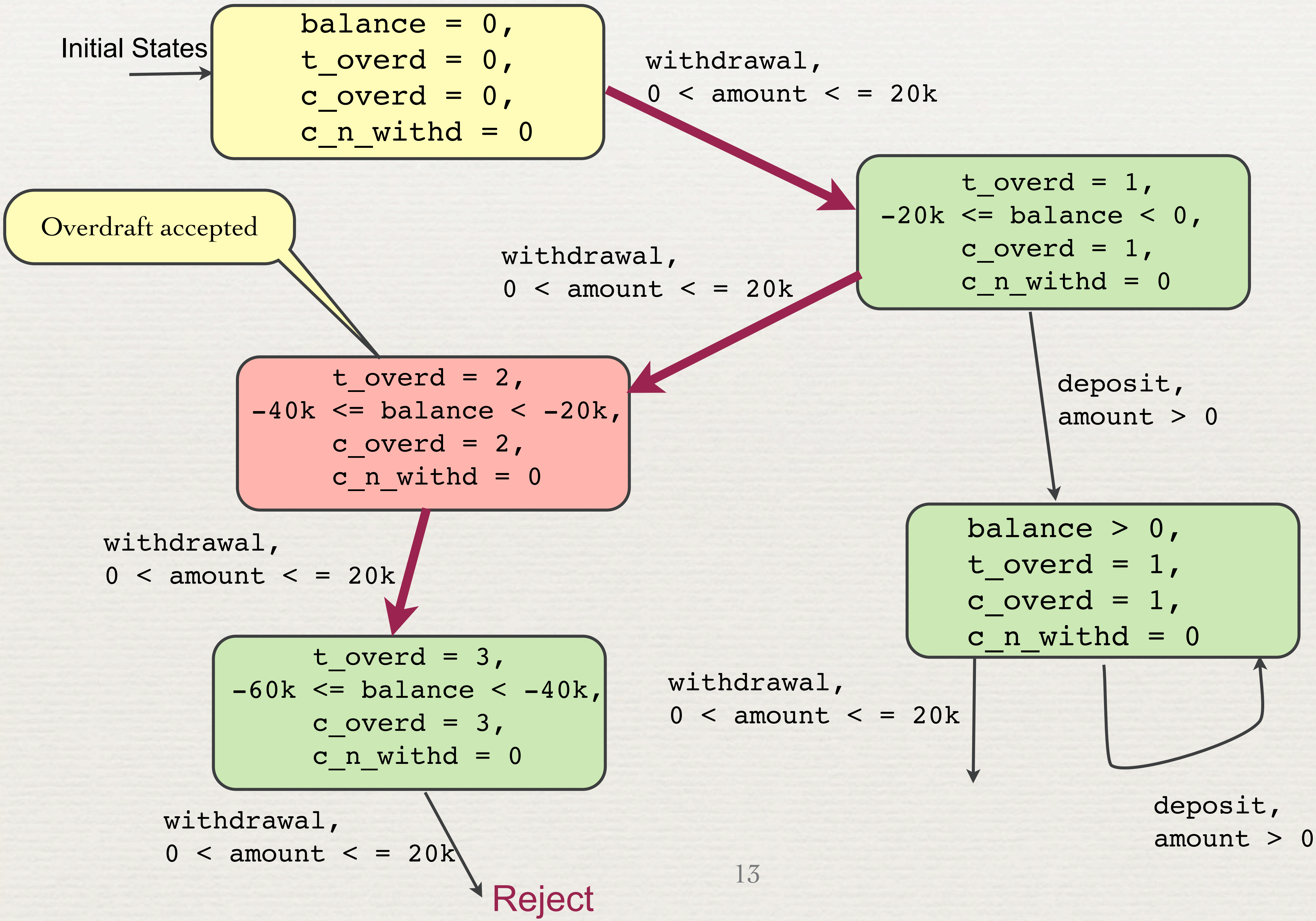


Answering Verification Properties:Reachability



Answering Verification Properties:Temporal Properties

Example: If an overdraft is rejected there is some overdraft that was accepted in the past.



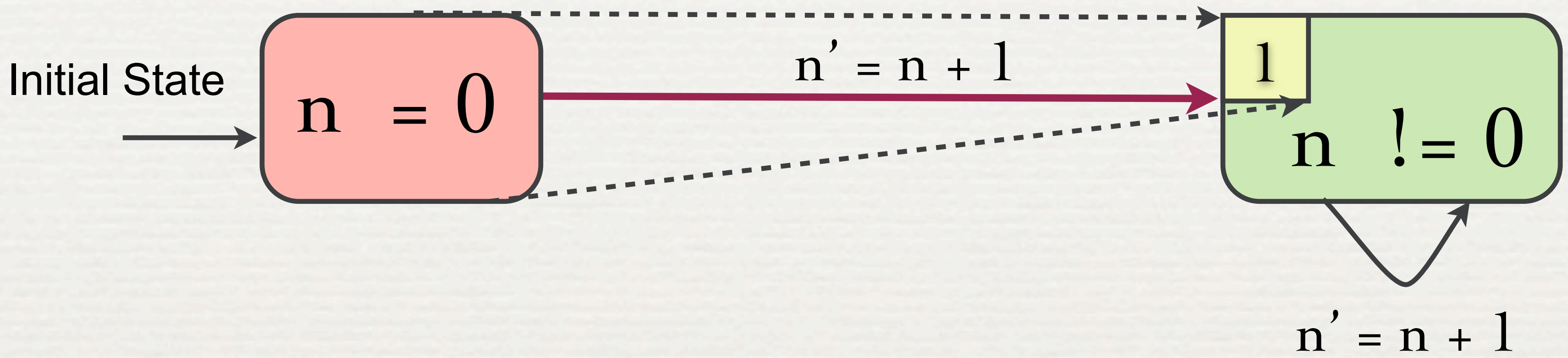
Our Algorithm: Summary

- Final partitioning satisfies following property:
  - If there is an edge between partitions P1 and P2, then every concrete state in P2 has a pre-image in P1.
- We can answer some temporal properties other than reachability.
  - Future work: To define the class of temporal properties that can be answered with final partitioning.
- The algorithm does not terminate on all counter automata.
  - Future work: To improve our algorithm to terminate on larger class of systems.

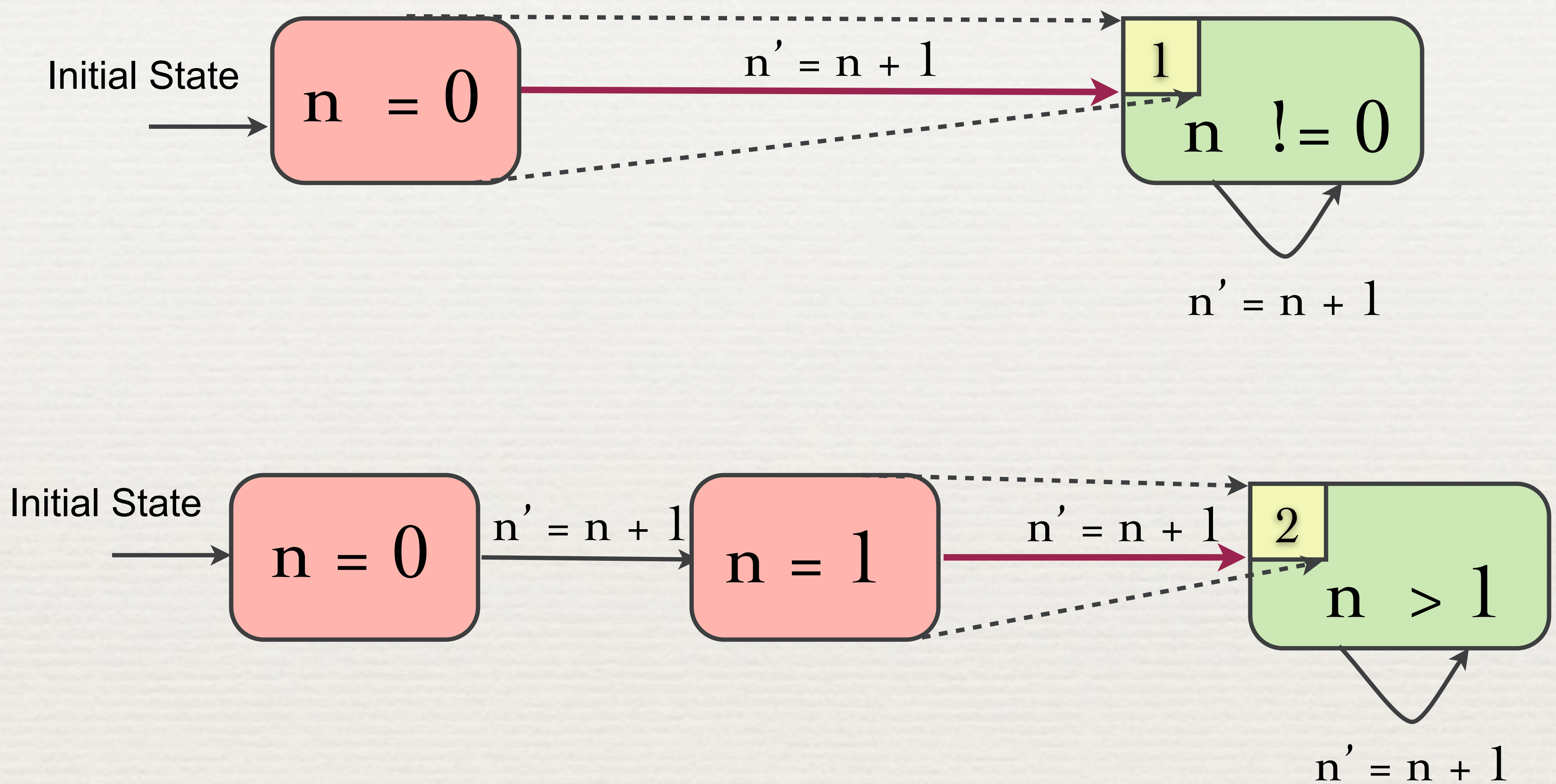
Our Algorithm: Proposed Improvements

- Algorithm can be tuned to terminate on larger class of counter automata using following changes:
  - To compute Image* instead of Image while refining on self loops on partitions.
  - To avoid partitioning of already reachable states.
  - To answer reachability for a given set of final states.
- These improvements will make our algorithm equivalent to symbolic model checking.

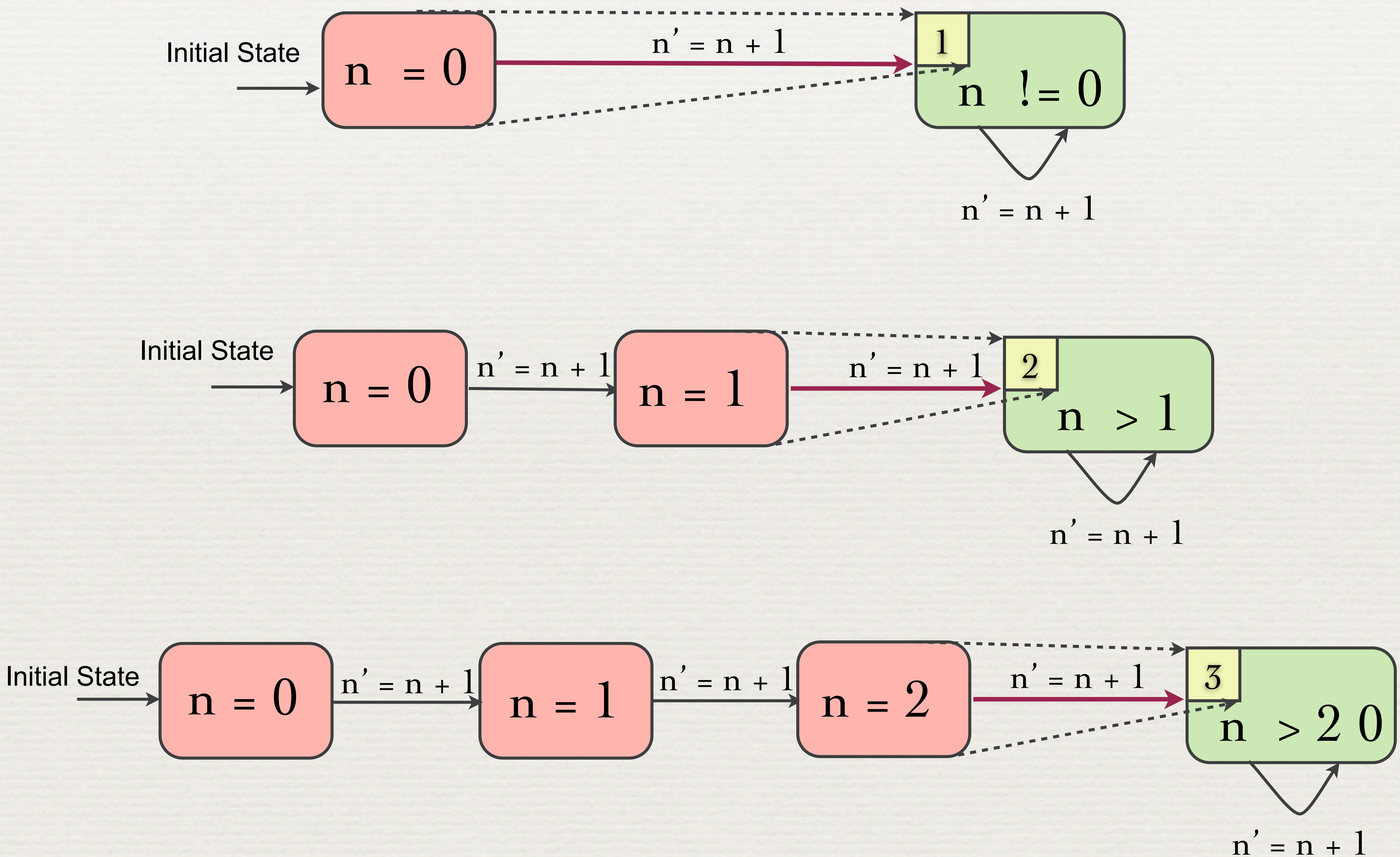
Example: Computing Image*

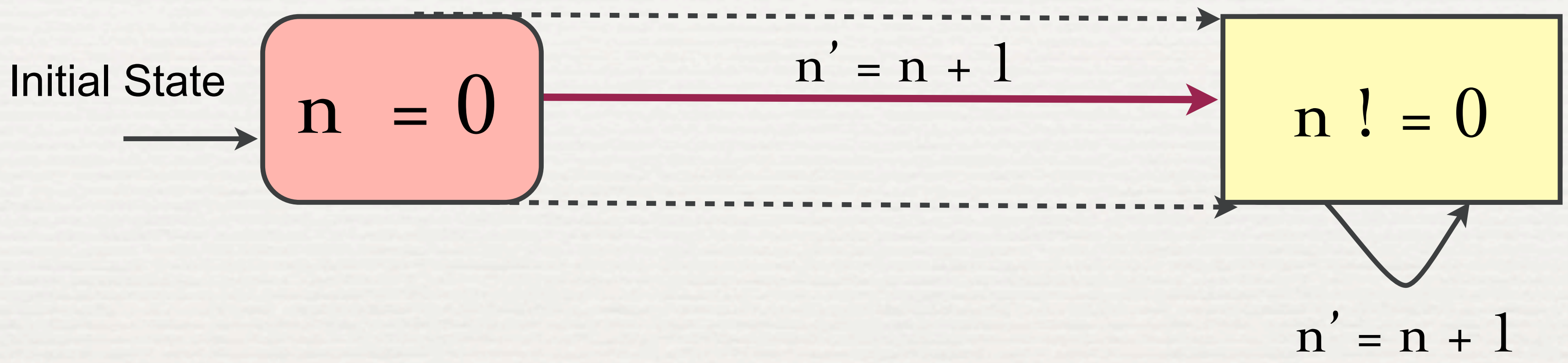


Example: Computing Image*



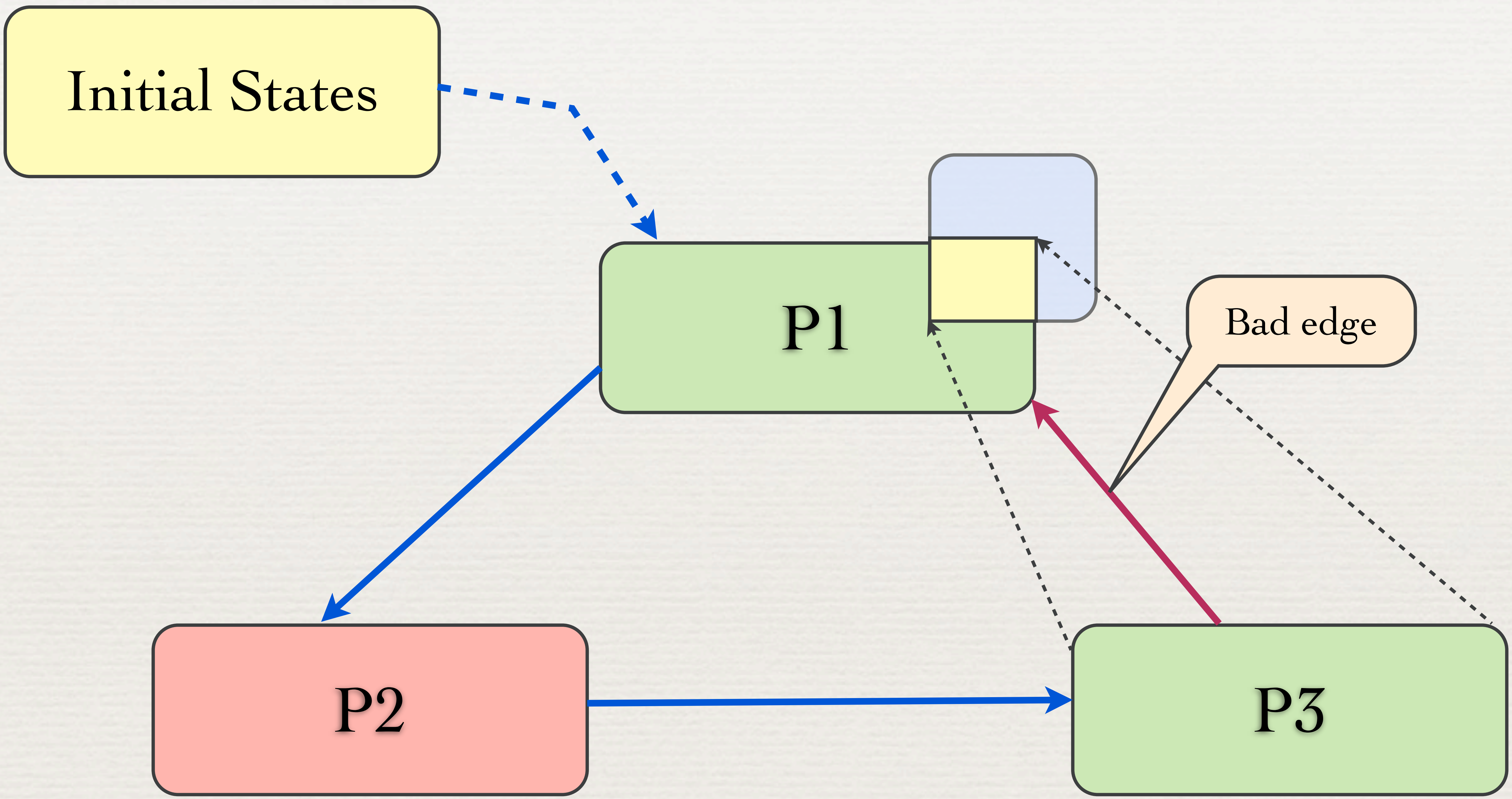
Example: Computing Image*



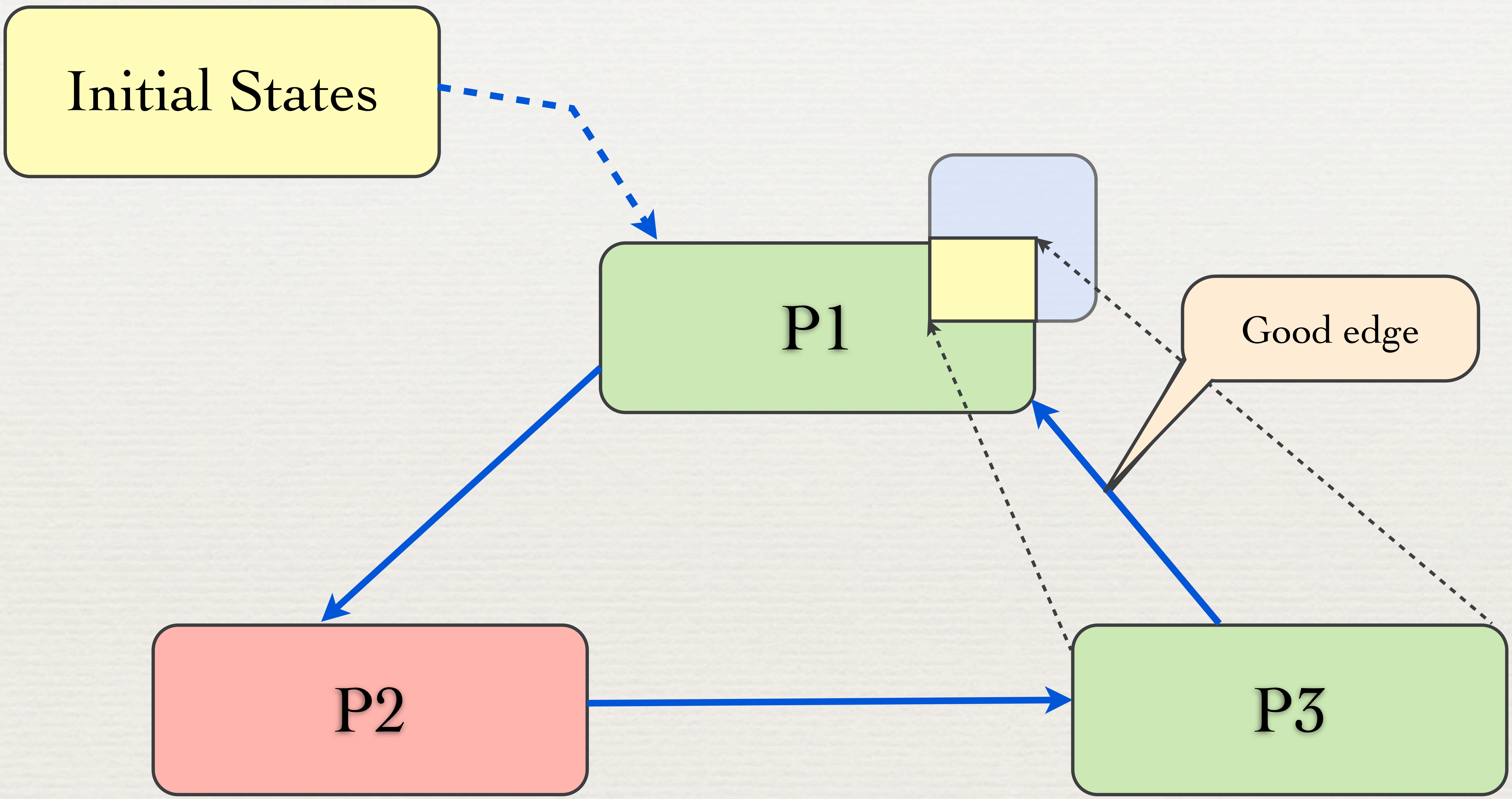


- We loose the information about the paths on which we could reach a particular state.

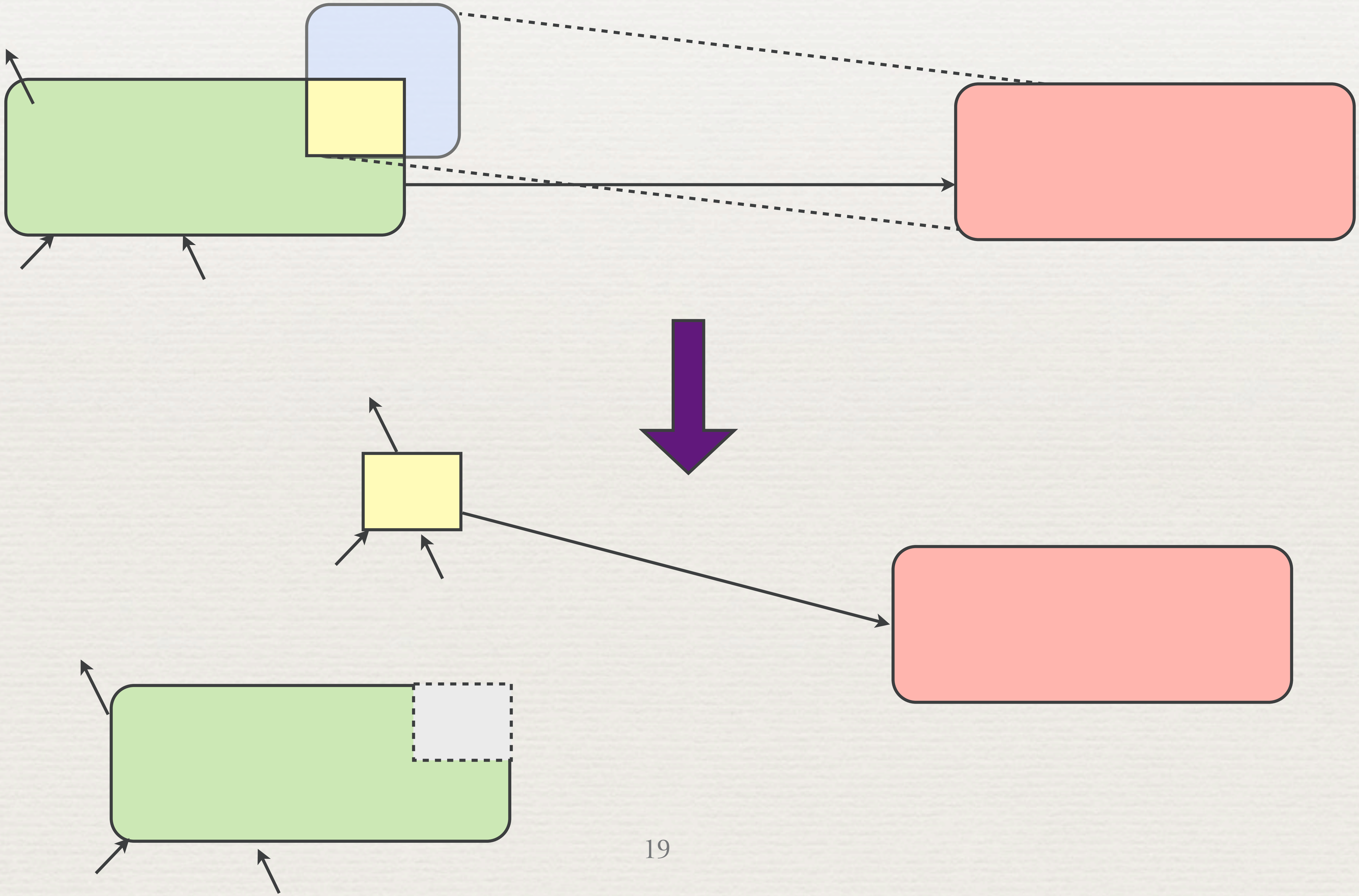
Avoid Partitioning of Already Reachable States



Avoid Partitioning of Already Reachable States



Backward Analysis



Answering properties using Backward Analysis

- Control state reachability: Given a transition system with a finite labeling for concrete states, we can only answer reachability of a label using our backward analysis algorithm.
- Backward algorithm terminates on some examples where forward algorithm fails to terminate, and vice versa.
- Forward algorithm:
  - *Must* analysis for reachability of a concrete state form initial state.
  - *May* analysis for reachability of a final state from given concrete state.
- Backward algorithm:
  - *May* analysis for reachability of a concrete state form initial state.
  - *Must* analysis for reachability of a final state from given concrete state.

In Comparison to Existing Algorithms

- Algorithms for hybrid automata proposed by Henzinger et. al.
  - Partitioning based
  - Backward analysis of the system
  - Answer control state reachability
- Synergy proposed by Bhargav S. Gulavani et. al.
  - Partitioning based
  - Backward analysis
  - Answers reachability of a given set of final states.
- Symbolic model checking of infinite state systems
  - Forward analysis
  - Set saturation based

Comparison to Ibarra et. al. (2000)

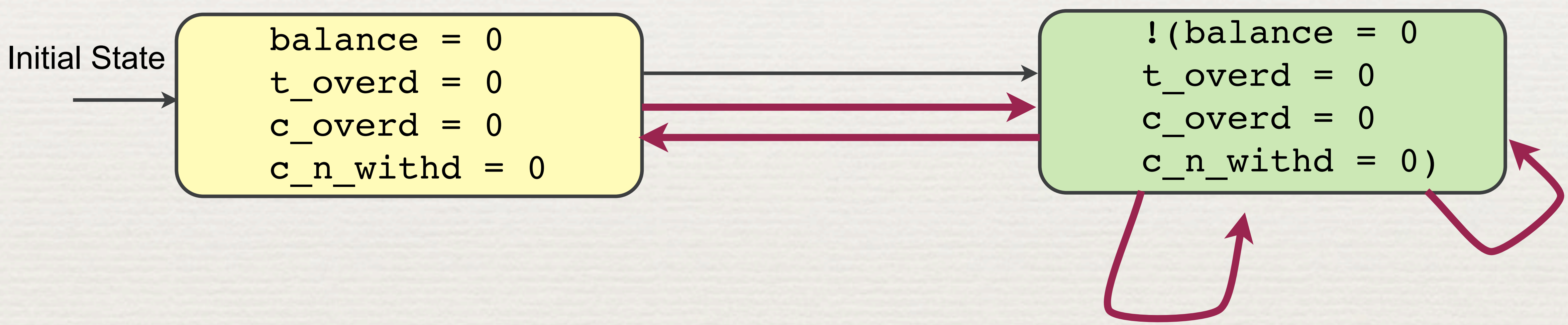
- Result: Emptiness, infiniteness, disjointness, containment and equivalence is decidable for reversal bounded counter machines.
- The banking example is not reversal bounded:
  - `balance` and `cn_withd` are not reversal bounded.
- Our algorithm doesn't terminate on all reversal bounded counter automata.
- Future work: To improve our algorithm so that it terminates on all reversal bounded counter automata.

Comparison to Comon et. al. (1998)

- Result: Reachability is decidable in flat counter automata.
- Counter automata for the banking example is not a flat.

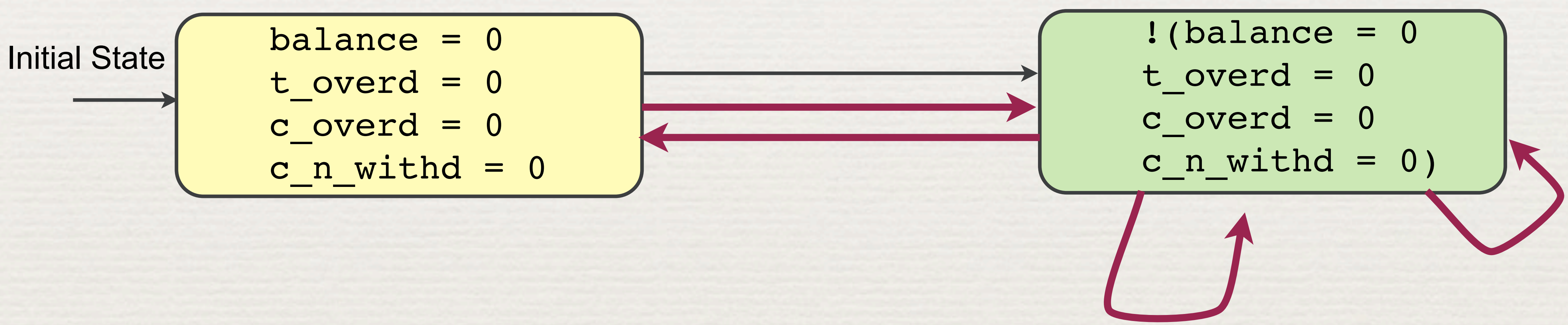
Comparison to Comon et. al. (1998)

- Result: Reachability is decidable in flat counter automata.
- Counter automata for the banking example is not a flat.



Comparison to Comon et. al. (1998)

- Result: Reachability is decidable in flat counter automata.
- Counter automata for the banking example is not a flat.



- The guard on edges is specified as conjunctions formulas of the form: x <= y + d, where x and y can be primed or unprimed counters and d is a constant.
- Future Work: Check if our algorithm terminates on all flat automata.

Comparison to Past Work: Summary

- The two papers define classes of counter automata for which reachability is decidable.
- Future Work:
  - Identify the sufficient conditions for termination of our algorithms.
  - To improve our algorithms to terminate on a class of counter automata that is superset of reversal bounded and flat automata.

Comparison to A. Finkel et. al. (1994)

- Result: Generalized results for answering control state reachability in well-structured infinite state transition systems
- Future work:
  - Check how well-structuredness of transitions systems is related to termination of our algorithms.

Thank You