

CS 520

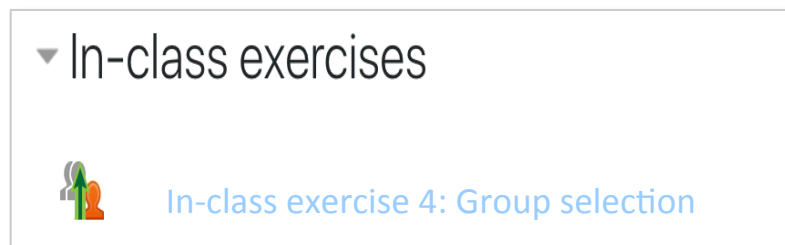
Theory and Practice of Software Engineering
Spring 2021

Model inference of processes

April 13, 2021

Thursday (April 15)

- Fourth (and last) in-class exercise
- On model inference of processes (today is a prelude with useful info)
- Form 3-, 4-, or 5- person teams
 - Use Moodle to self-select a team; open from today until Thursday at 9 PM



Problem

- Missing or inaccurate system documentation makes it challenging to understand the intended system behaviors
- Complex logs of unintended system behaviors makes it difficult to debug them

Goal

- Take as input a set of observed **traces** (usually represented as sequences of **events**) of a given system
- Automatically produce an **inferred model** (often represented as an FSA) that must accept all of the observed traces
 - May also accept some unobserved traces

Model inference tool: Architecture

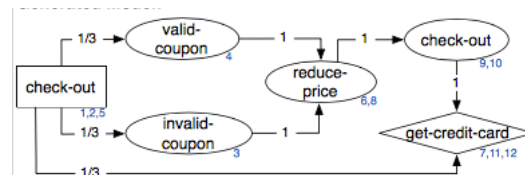
[Synoptic]

Sequences of events
(often method calls)
[Log + Regular expressions]

```
1|74.15.155.103 [06/Jan/2011:07:24:13] "GET HTTP/1.1 /check-out.php"  
2|13.15.232.201 [06/Jan/2011:07:24:19] "GET HTTP/1.1 /check-out.php"  
3|13.15.232.201 [06/Jan/2011:07:25:33] "GET HTTP/1.1 /invalid-coupon.php"  
4|74.15.155.103 [06/Jan/2011:07:27:05] "GET HTTP/1.1 /invalid-coupon.php"  
5|74.15.155.199 [06/Jan/2011:07:28:43] "GET HTTP/1.1 /check-out.php"  
6|74.15.155.103 [06/Jan/2011:07:28:14] "GET HTTP/1.1 /reduce-price.php"  
7|74.15.155.199 [06/Jan/2011:07:29:02] "GET HTTP/1.1 /get-credit-card.php"  
8|13.15.232.201 [06/Jan/2011:07:30:22] "GET HTTP/1.1 /reduce-price.php"  
9|74.15.155.103 [06/Jan/2011:07:30:55] "GET HTTP/1.1 /check-out.php"  
0|13.15.232.201 [06/Jan/2011:07:31:17] "GET HTTP/1.1 /check-out.php"  
1|13.15.232.201 [06/Jan/2011:07:31:20] "GET HTTP/1.1 /get-credit-card.php"  
2|74.15.155.103 [06/Jan/2011:07:31:44] "GET HTTP/1.1 /get-credit-card.php"
```

Model inference algorithm
[Bisim algorithm]

Inferred model
[Conceptually an FSA]



Shopping cart: API (written in PHP)

- invalid-coupon
- valid-coupon
- reduce-price
- check-out
- get-credit-card

BisimH algorithm

1. Extract a **trace graph** from a given **log** using the **regular expressions**
2. Mine **invariants** from that **trace graph**
3. Create **initial inferred model** by partitioning the **trace graph**
4. While (**current inferred model** violates **invariants**)
 1. Generate **counterexample path** through the current inferred model illustrating a violation of a given invariant
 2. Refine current inferred model based on that counterexample path to produce next inferred model satisfying that invariant
5. Coarsen current inferred model to produce final inferred model

BisimH algorithm

1. Extract a **trace graph** from a given **log** using the **regular expressions**
2. Mine **invariants** from the **trace graph**
3. Create **initial inferred model** by partitioning **trace graph**
4. While (**current inferred model** violates **invariants**)
 1. Generate **counterexample path** through the current inferred model illustrating a violation of a given invariant
 2. Refine current inferred model based on that counterexample path to produce next inferred model satisfying that invariant
5. Coarsen current inferred model to produce final inferred model

1. Extract: Log and regular expressions

- Log, e.g.,

Line: 74.15.155.103 [06/Jan/2011:07:24:13] "GET HTTP/1.1/check-out.php"

- Regular expression(s), e.g.,

Line parsing: (?<ip>).+/(?<TYPE>.+).php

- Special events for INITIAL and TERMINAL. Each **event** specified as a triple <trace ID, timestamp, event type>, e.g., <74.15.155.103, 06/Jan/2011:07:24:13, check-out>

1. Extract: Trace graph

- Each **event** specified as a triple
<trace ID, timestamp, event type>
 - Also special events for INITIAL and TERMINAL
- Each **trace** is a linear graph where:
 - Each event corresponds to a vertex
 - The total ordering among the events (specified by their timestamps) corresponds to the edges among the nodes
- A **trace graph** is the union of the set of traces

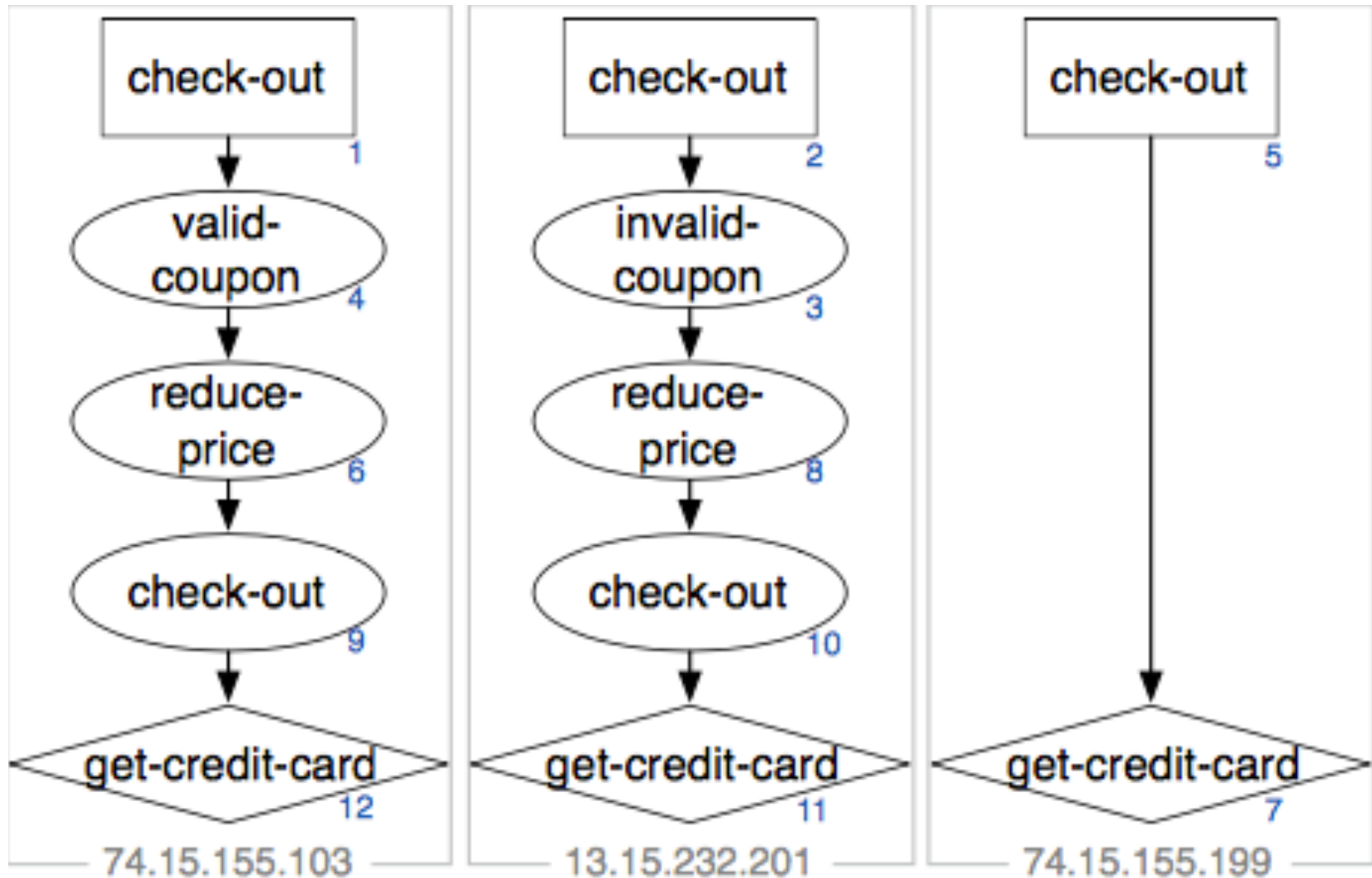
Shopping cart: Inputs

Log:

1	74.15.155.103	[06/Jan/2011:07:24:13]	"GET HTTP/1.1 /check-out.php"
2	13.15.232.201	[06/Jan/2011:07:24:19]	"GET HTTP/1.1 /check-out.php"
3	13.15.232.201	[06/Jan/2011:07:25:33]	"GET HTTP/1.1 /invalid-coupon.php"
4	74.15.155.103	[06/Jan/2011:07:27:05]	"GET HTTP/1.1 /valid-coupon.php"
5	74.15.155.199	[06/Jan/2011:07:28:43]	"GET HTTP/1.1 /check-out.php"
6	74.15.155.103	[06/Jan/2011:07:28:14]	"GET HTTP/1.1 /reduce-price.php"
7	74.15.155.199	[06/Jan/2011:07:29:02]	"GET HTTP/1.1 /get-credit-card.php"
8	13.15.232.201	[06/Jan/2011:07:30:22]	"GET HTTP/1.1 /reduce-price.php"
9	74.15.155.103	[06/Jan/2011:07:30:55]	"GET HTTP/1.1 /check-out.php"
10	13.15.232.201	[06/Jan/2011:07:31:17]	"GET HTTP/1.1 /check-out.php"
11	13.15.232.201	[06/Jan/2011:07:31:20]	"GET HTTP/1.1 /get-credit-card.php"
12	74.15.155.103	[06/Jan/2011:07:31:44]	"GET HTTP/1.1 /get-credit-card.php"

Regular Expressions: Line parsing: "(?<ip>).+/(?<TYPE>.+).php"
Trace mapping: "\k<ip>"

Shopping cart: Trace graph



BisimH algorithm

1. Extract a **trace graph** from a given **log** using the **regular expressions**
2. Mine **invariants** from the **trace graph**
3. Create **initial inferred model** by partitioning **trace graph**
4. While (**current inferred model** violates **invariants**)
 1. Generate **counterexample path** through the current inferred model illustrating a violation of a given invariant
 2. Refine current inferred model based on that counterexample path to produce next inferred model satisfying that invariant
5. Coarsen current inferred model to produce final inferred model

2. Mine invariants

- Capture temporal relationships between **event types** in a given trace graph
 - a Always Followed by b ($a \rightarrow b$)
e.g., INITIAL \rightarrow check-out
 - a Never Followed by b ($a \nrightarrow b$)
e.g., valid-coupon \nrightarrow invalid-coupon
 - a Always Precedes b ($a \leftarrow b$)
e.g., check-out \leftarrow get-credit-card
- Must be satisfied by all of the potential traces through that graph

2. Mine invariants

- Capture temporal relationships between **event types** in a given trace graph
 - a Always Followed by b ($a \rightarrow b$)
e.g., INITIAL \rightarrow check-out
 - a Never Followed by b ($a \nrightarrow b$)
e.g., valid-coupon \nrightarrow invalid-coupon
 - a Always Precedes b ($a \leftarrow b$)
e.g., check-out \leftarrow get-credit-card
- Must be satisfied by all of the potential traces through that graph

Use the Dwyer et al.
property patterns

Shopping cart: Invariants

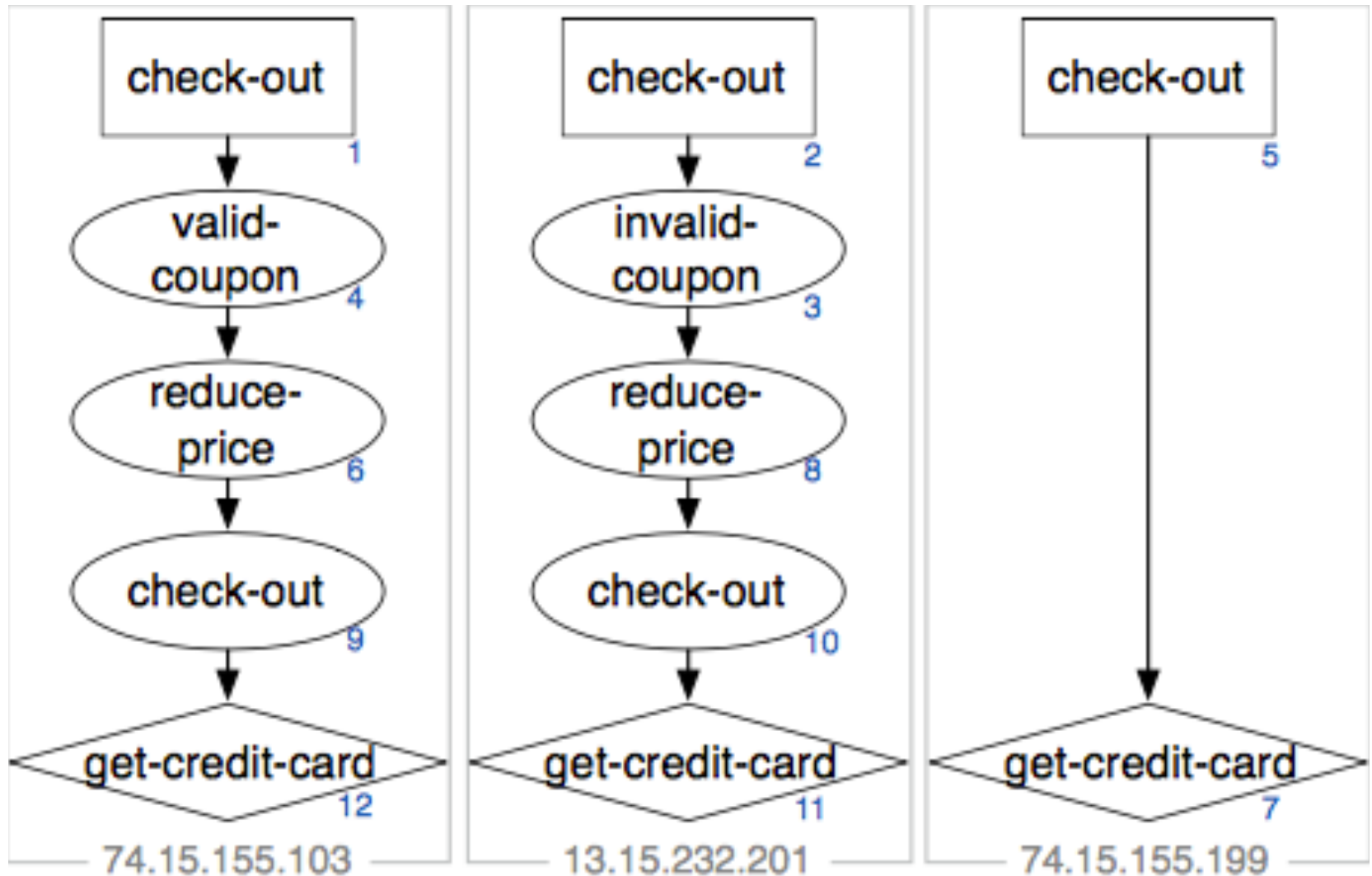
Show invariants as text



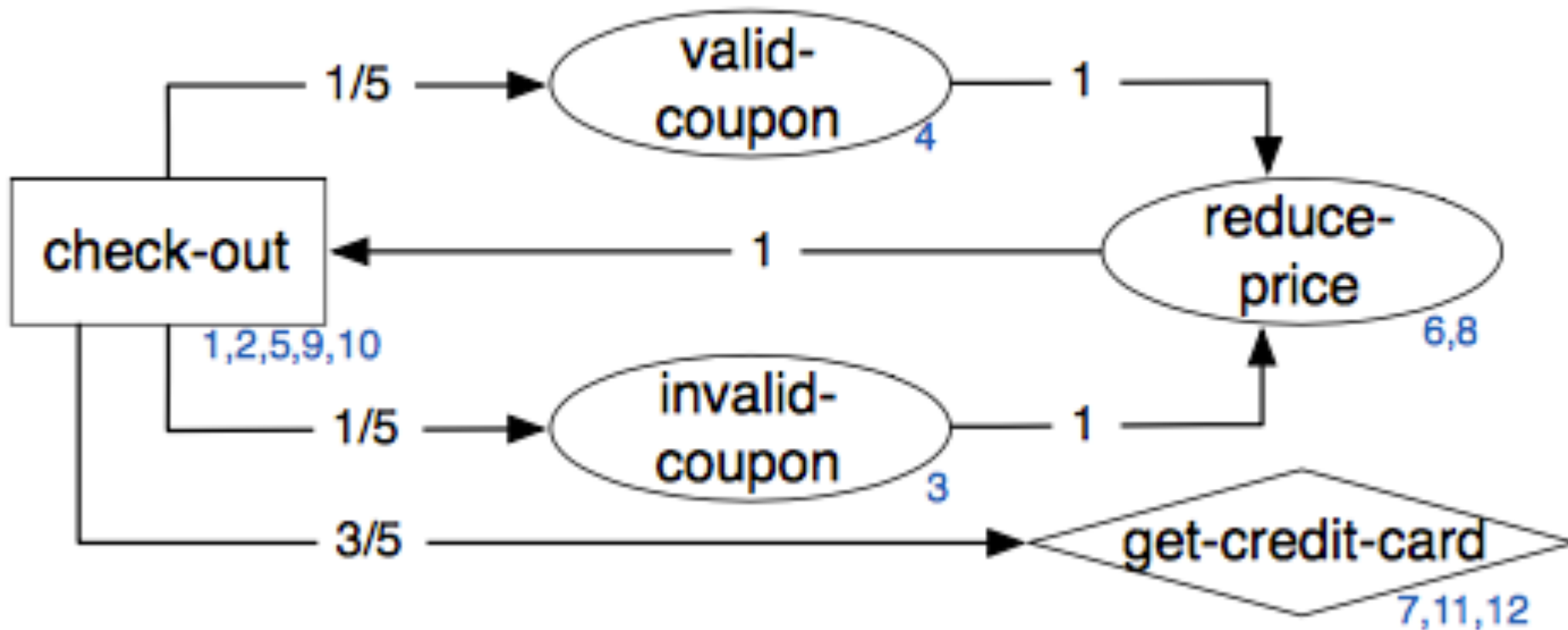
BisimH algorithm

1. Extract a **trace graph** from a given **log** using the **regular expressions**
2. Mine **invariants** from the **trace graph**
3. Create **initial inferred model** by partitioning **trace graph**
4. While (**current inferred model** violates **invariants**)
 1. Generate **counterexample path** through the current inferred model illustrating a violation of a given invariant
 2. Refine current inferred model based on that counterexample path to produce next inferred model satisfying that invariant
5. Coarsen current inferred model to produce final inferred model

Shopping cart: Trace graph



Shopping cart: Initial model



BisimH algorithm

1. Extract a **trace graph** from a given **log** using the **regular expressions**
2. Mine **invariants** from the **trace graph**
3. Create **initial inferred model** by partitioning **trace graph**
4. While (**current inferred model** violates **invariants**)
 1. Generate **counterexample path** through the current inferred model illustrating a violation of a given invariant
 2. Refine current inferred model based on that counterexample path to produce next inferred model satisfying that invariant
5. Coarsen current inferred model to produce final inferred model

BisimH algorithm

1. Extract a **trace graph** from a given **log** using the **regular expressions**
2. Mine **invariants** from the **trace graph**
3. Create **initial inferred model** by partitioning **trace graph**
4. While (**current inferred model** violates **invariants**)
 1. Generate **counterexample path** through the current inferred model illustrating a violation of a given invariant
 2. Refine current inferred model based on that counterexample path to produce next inferred model satisfying that invariant
5. Coarsen current inferred model to produce **final inferred model**

Use counterexample guided abstraction refinement (CEGAR)

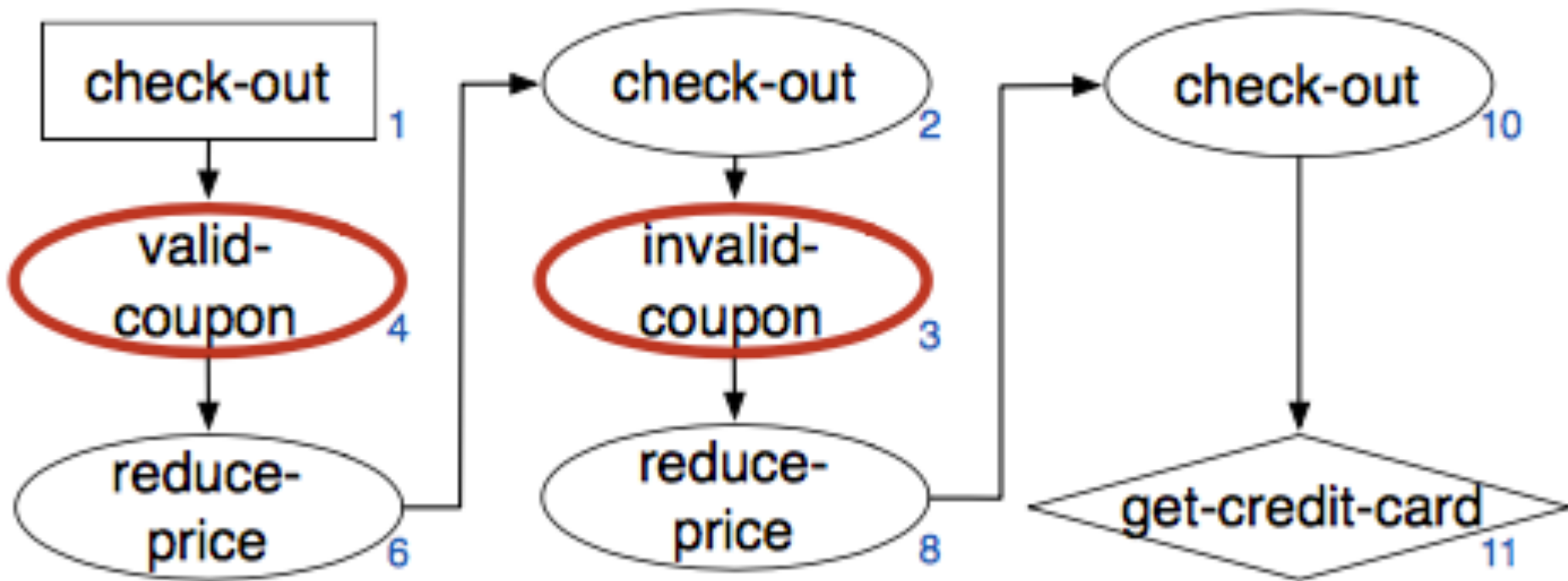
4. CEGAR:

Generate counterexample path

- Employ a model checker to determine if all potential paths through the **current inferred model** satisfy the **invariants**
 - If so, report satisfied
 - If not, report violated and a **counterexample path** illustrating a violation of one of the invariants

Shopping cart: Counterexample path

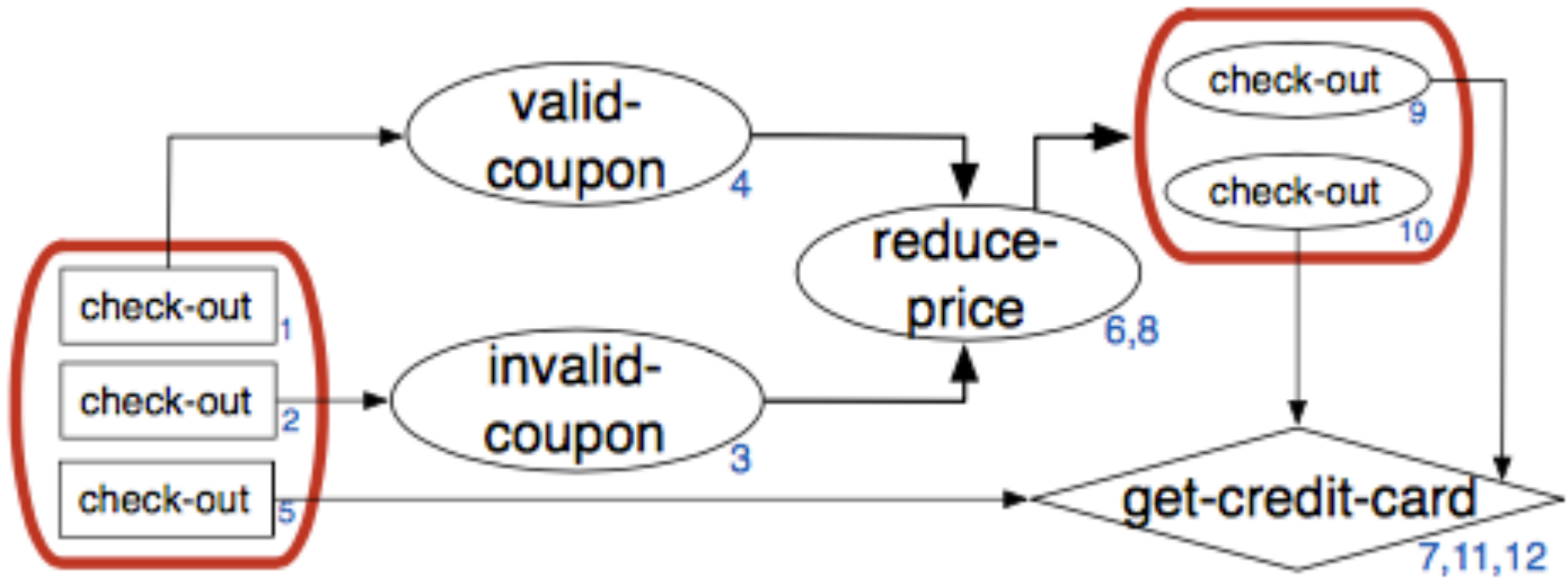
Violates invariant: valid-coupon \nrightarrow invalid-coupon



4. CEGAR: Refine

1. Refine the current inferred model based on the violated invariant
 - Heuristically select a vertex relevant to that invariant
2. Create the refined inferred model that satisfies that invariant
 - Split the selected vertex into multiple vertices

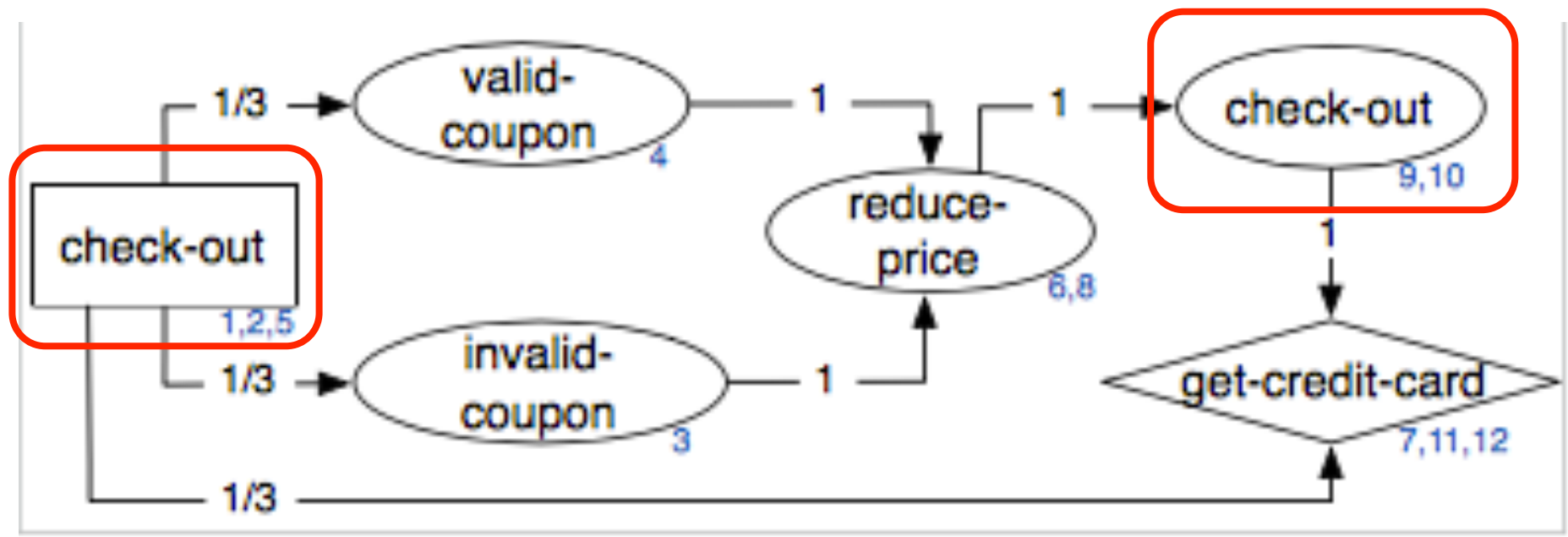
Shopping cart: Refined model



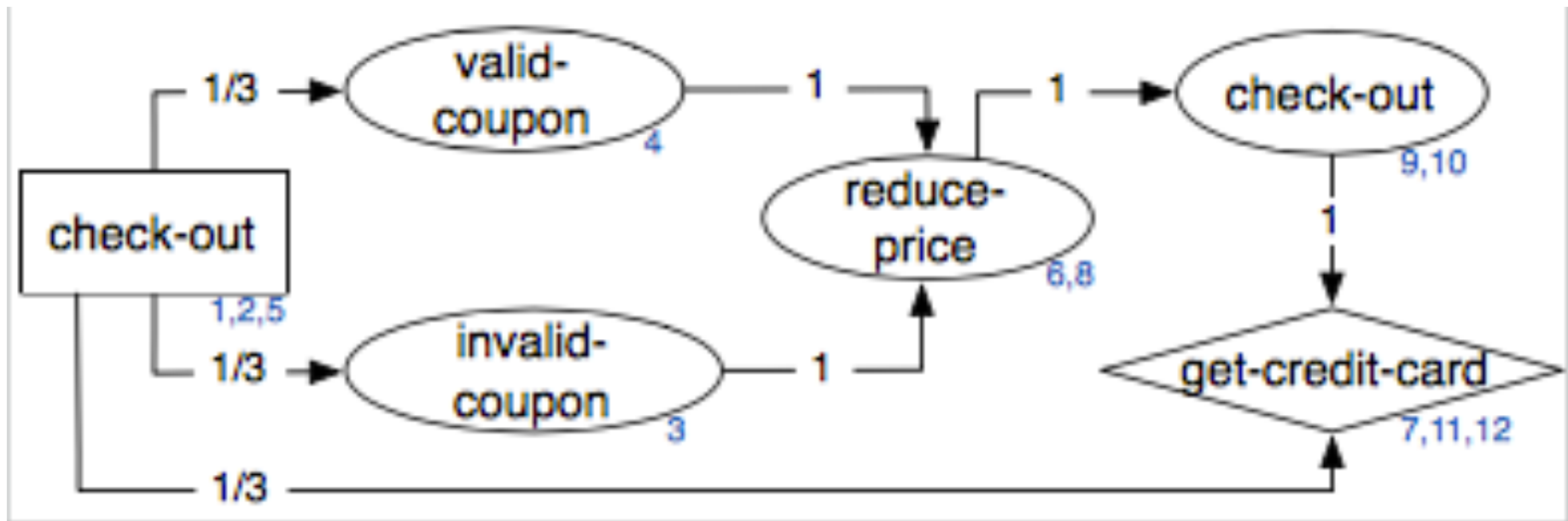
5. Coarsen

- Coarsen the refined inferred model
 - Search for vertices that didn't need to be split by the refine heuristic and merge them
- The coarsened inferred model satisfies all of the invariants. (It is the final inferred model.)

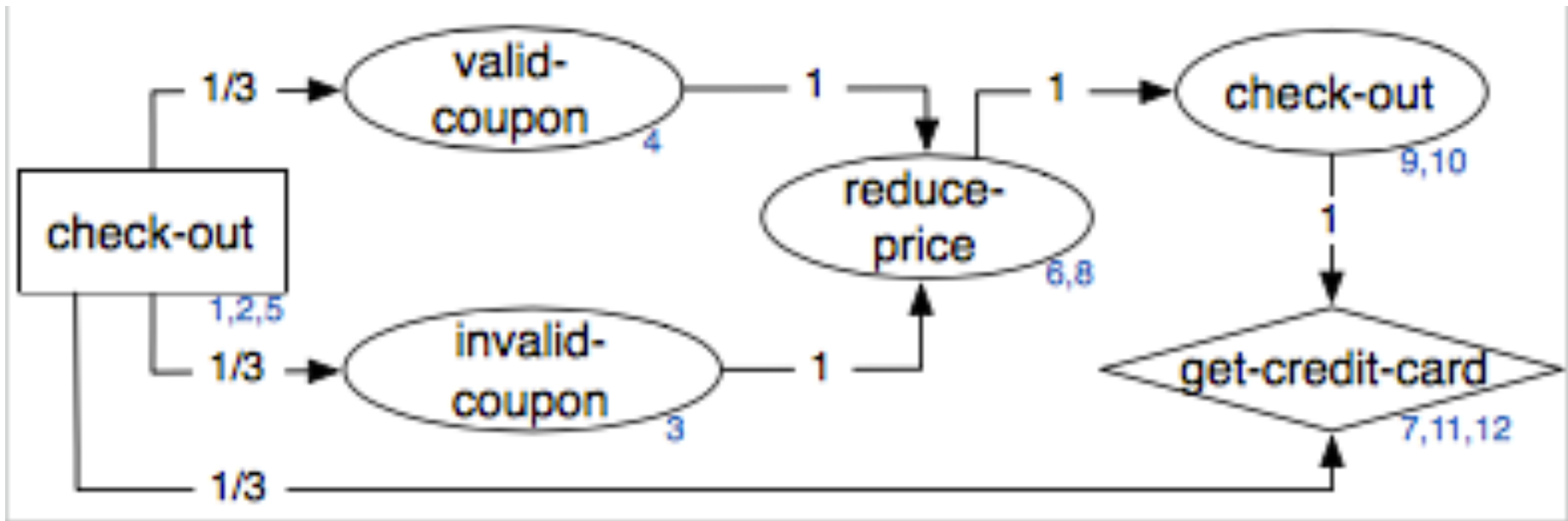
Shopping cart: Coarsened model



Shopping cart: Final model

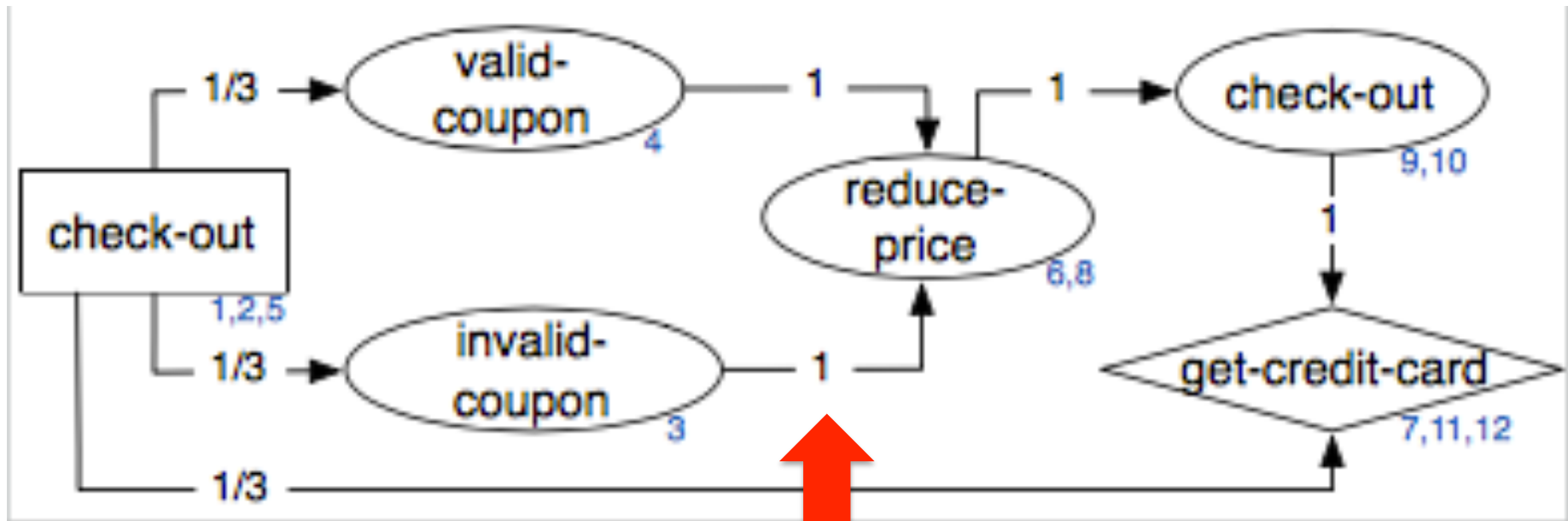


Shopping cart: Final model



Does the inferred model illustrate any unintended behavior (i.e. a bug)?

Shopping cart: Final model



Does the inferred model illustrate any unintended behavior (i.e. a bug)? **YES**

Challenges for model inference

- How does the selection of the event sequence set affect the final model? affect performance?
- How does the parameterization of the model inference algorithm affect the final model?
- How do you compare/contrast (or diff) two inferred models?
- How to support other high-level language features such as concurrency, real-time constraints, ...?

Learning from system traces

- Logs, e.g., Synoptic, InvariMint

https://www.cs.ubc.ca/~bestchai/papersinvarimint_icse13.pdf

- Counterexample paths, e.g., libalf: Automata learning framework

<http://libalf.informatik.rwth-aachen.de>

- Usage scenarios, e.g.,

<https://dl.acm.org/doi/pdf10.1145/1656250.1656252>

- ...