Amorphous Slicing for EFSMs

PLID'08

David Clark
King's College London

Tuesday 15 July 2008

Collaborators

- Kalli Androutsopoulos
- Dave Binkley
- Mark Harman
- Laurie Tratt

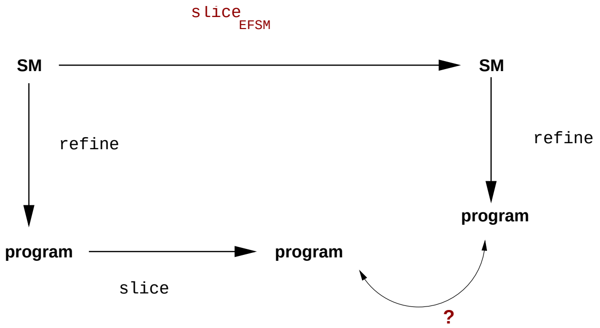
Talk outline

- Slicing state based models
- existing work
- Definition of EFSMs
- Korel's ATM example
- Data and Control Dependence
- Slicing algorithm
- Applying the algorithm to Korel's ATM example
- Comments, issues and conclusions

Slicing State-based Models

- State-based model: Statecharts/Finite State Machines and variants.
- Extended Finite State Machines (EFSMs).
- FSMs extended with a store and store update actions on transitions.
- programming language with abstract, flexible control structures and non-determinism.

Some Questions



Existing work on Slicing State-based Models

- Korel, Singh, Tahat & Vaysburg: "Slicing State-based Models" [2003].
- Labbé and Gallois [2001?] Brief paper giving some definitions for slicing communicating extended automata. CD close to ranganath et al. NTSCD
- Fox and Luangsodai [2006] **and** and **or** state dependencies in state charts
- Labbé and Lapitre [2007] report on the CARVER tool
- Scaeffer and Poetzsch-Heffter

First is our jumping off point.

EFSM: formal definition

- A set of States, S.
- A set of Transitions, T.
- A set of Events, Ev.
- A store represented by a set of local variables, Var.
- A set of Phrases, P where P has grammar:

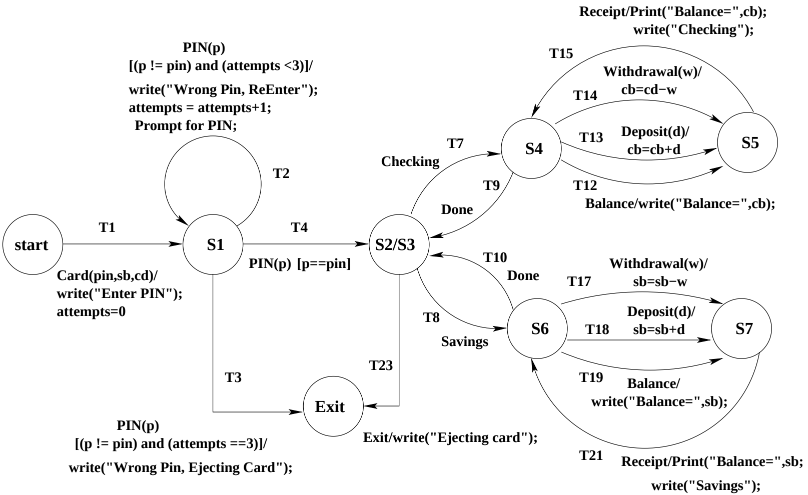
$$C \in \mathrm{Com} \qquad x \in \mathrm{Var} \qquad E \in \mathrm{Exp} \qquad B \in \mathrm{BExp} \qquad n \in \mathbb{Z}$$

$$\begin{aligned}
P &::= C \mid B \\
C &::= x := E \mid C_1 ; C_2 \\
E &::= x \mid n \mid E_1 + E_2 \mid E_1 - E_2 \mid E_1 * E_2 \quad \mid \\
B &::= \neg B \mid B_1 \wedge B_2 \mid E_1 < E_2 \mid E_1 == E_2
\end{aligned}$$

EFSM: formal definition

- A Transition, $t \in T$, is given by
  - A source state $src(t) \in S$.
  - A label, $lbl(t)$, where $lbl(t)$ has the form $e_1[b]/e_2.c$ where $e_1, e_2 \in Ev$, $b \in B$, $c \in C$ and all parts of the label are optional.
  - A target state $tgt(t) \in S$.
- States of S are atomic.
- Machines are possibly non-deterministic.
- Actions can involve store updates or generation of events or both.
- Logical guards, where they exist, refer to the store

Example

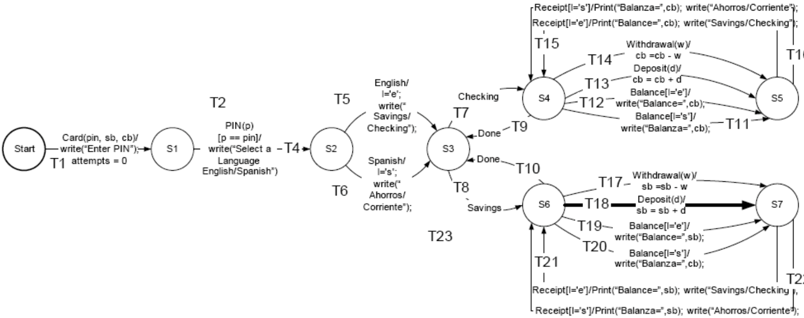


PIN(p)
[(p != pin) and (attempts <3)]/
write("Wrong Pin, ReEnter");
attempts = attempts+1;
Prompt for PIN;
T2
start
T1
S1
T4
S2/S3
Card(pin,sb,cd)/
write("Enter PIN");
attempts=0
PIN(p) [p==pin]
T3
Exit
T23
PIN(p)
[(p != pin) and (attempts ==3)]/
write("Wrong Pin, Ejecting Card");
Exit/write("Ejecting card");
Checking
T7
T9
Done
S4
S5
Receipt/Print("Balance=",cb);
write("Checking");
T15
Withdrawal(w)/
cb=cd−w
T14
T13
Deposit(d)/
cb=cb+d
T12
Balance/write("Balance=",cb);
T10
Done
T8
Savings
S6
S7
Withdrawal(w)/
sb=sb−w
T17
Deposit(d)/
sb=sb+d
T18
T19
Balance/
write("Balance=",sb);
T21
Receipt/Print("Balance=",sb;
write("Savings");

Problem with Korel et alia Definition of Slicing

- Closely follows program slicing: assumes exit state (final state)
- Slice on variable *sb* on transition T_{18} in first example
- The slice is clearly too large
- Their definition simply removes the two transitions and one state which are not on any path from start that reaches T_{18}
- Their solution is to shrink the number of nodes by introducing non-determinism

ATM EFSM slice



Data Dependence

Let $D(T)$ be the set of variables defined on transition T and $U(T)$ be the set of variables used on transition T.

Definition (data dependence for EFSMs)

[Korel et alia] There is data dependence between transitions T_i and T_k w.r.t. variable x if

- $x \in D(T_i)$,
- $x \in U(T_k)$, and
- there is a path (transition sequence) from T_i to T_k along which x is not defined.

Reactive Systems and Control Dependence

- "A New Foundation for Control Dependence and Slicing for Modern Program Structures" by Ranganath, Amtoft, Banerjee, Hatcliffe and Dwyer [2005 (and later versions)]
- modern: non-terminating threads, exceptions
- apply to EFSMs; deal with no exit state or many exit states, i.e. potentially non-terminating reactive system
- EFSMs: dependence between transitions, rather than between nodes

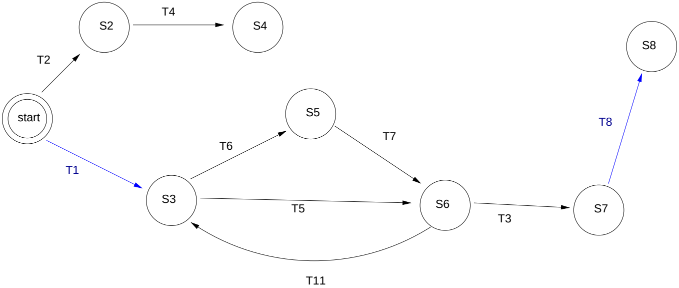
Control Sinks

- A **control sink**, $\mathcal{K}$, for an EFSM is a set of transitions that form a strongly connected component such that, for each transition t in $\mathcal{K}$ each successor of t is in $\mathcal{K}$.
- NB: framed using transitions rather than nodes.
- A **maximal path** is any path in an EFSM that terminates in a final transition or is infinite.
- A set of **sink-bounded paths** in an EFSM from a transition T, SinkPaths(T), contains all maximal paths π from T with the property that there exists a control sink $\mathcal{K}$ such that
  1. π contains transition T_s from $\mathcal{K}$;
  2. If π is infinite then all transitions in $\mathcal{K}$ occur infinitely often.

Non-Termination Insensitive Control Dependency

- In an EFSM, a transition T_j is (directly) control dependent on a transition T_i if and only if T_i has at least one sibling T_k such that
    1. For all paths $\pi \in \text{SinkPaths}(T_i)$, the source of T_j belongs to π;
    2. there exists a path $\pi \in \text{SinkPaths}(T_k)$ such that the source node of T_j does not belong to π.
- Corresponds to "traditional" slicing in so far as it allows the slice to omit loops present in the original graph
- **Proposition**: for EFSMs with a unique end node T_i is control dependent on T_j in Korel's sense iff T_i is NTICD on T_j.

Example



T_8 is NTICD on T_1.

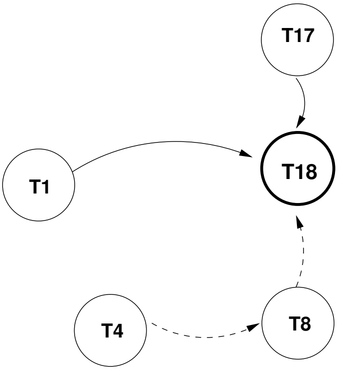
shrinking the slice

- We set out to make the slice as small as possible
- Resulting graph is not a subgraph of the original
- Natural result in the graph world?

Slicing Algorithm

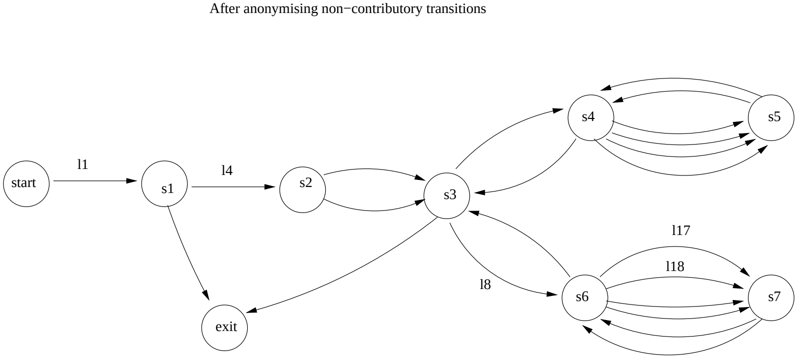
- Transition of interest is T_i
- Use *data dependence* and *control dependence* definitions to construct the (transitively closed) Machine Dependency Graph (MDG) for T_i
- Mark the transitions appearing in the MDG
- Remove unmarked transitions
  - We use exactly the same approach as the construction used in the "Silent Moves" lemma for FSAs, treating unmarked transitions as "silent moves"
- Collect garbage
- Merge states
  - Another standard FSA minimisation algorithm: two states are merged if they have identical outward transitions; this process is repeated until no further states can be merged

MDG for Korel et al. example

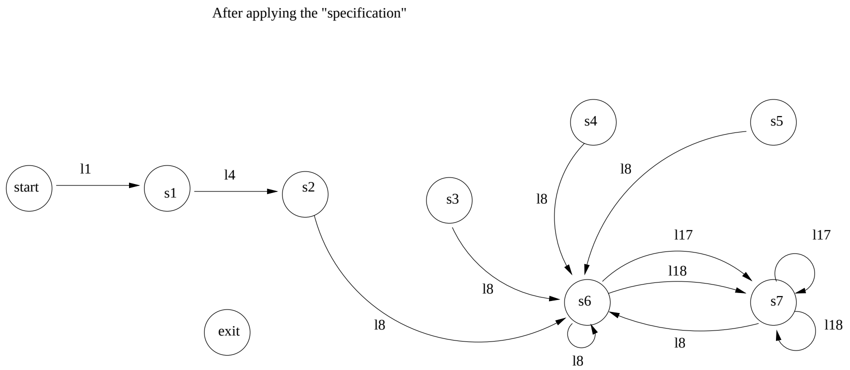


MDG for T18

After marking transitions

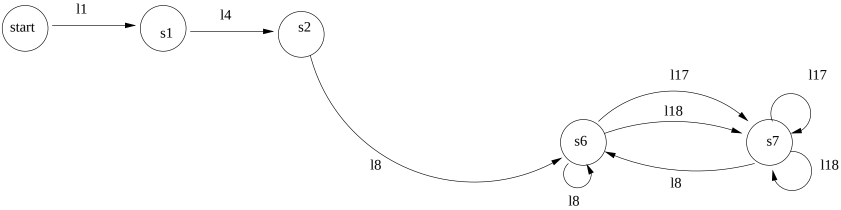


After removing “silent moves”



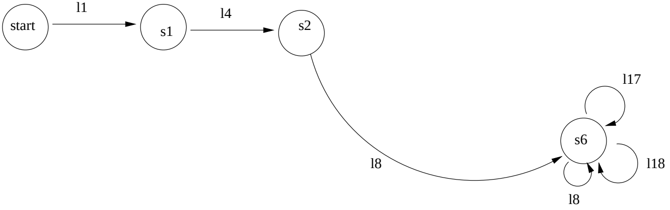
After garbage collection

After garbage collection



After merging states

After merging



Some comments

- Dubbed "Amorphous slicing" (although no equivalent to program transformation used)
- Semantic correctness via simulation of paths through transitions of interest in the original by paths in the slice + "silent moves lemma" + state merging lemma.
- Currently building EFSM slicing tool
- Currently building slice animation tool
- have another paper which classifies different slicing approaches for these models and puts them into a general framework. Currently unpublished.

Some Issues

- Fairness condition implies no control dependencies inside a control sink
- **Proposition**: relaxing the fairness condition does not change dependencies outside of the control sinks.
- Investigating different ways of finding dependencies within control sinks
- Silent moves removal algorithm does not always shrink the graph
- Recently became aware of Torben Amtoft's paper on slicing using NTICD

Conclusions

- Have realised our initial aim of devising a slicing algorithm which (at least some times) produces small slices
- Improves comprehensibility in some ways but needs an animation tool for users to relate to the original graph easily
- Plenty of ongoing issues to resolve: a potentially large research area
- Research is now part of the (EPSRC) project Slicing state based Models (SLIM), employing two RAs
- Web page *slim.dcs.kcl.ac.uk*
- Soon: technical report at *www.dcs.kcl.ac.uk/technical-reports/*