Machine extraction

Simon Thompson

Göteborg, August 2011

1 Overview

- Formalise process of traces to QSM state machine
- Formalise processes from QSM state machine to QuickCheck FSM.

2 Traces to QSM state machine

2.1 Basic QSM

Definition of concrete traces:

```
item  ::= {module, function, arguments, result}
trace ::= {pos, item*} | {neg, item*}
```

If T is a set of traces then `t` is a positive trace if $\{$`pos,t`$\} \in T$; similarly for negative traces.

An abstraction function is a function

α :: $\{$`module, function, arguments, result`$\}$ `->` $\mathcal{A}$

for some type $\mathcal{A}$.. The *abstract traces* are given by $\{$`pos,map(abs,t)`$\}$ and $\{$`neg,map(abs,t)`$\}$.

A set T of traces is *consistent* iff for all $\{$`neg,t`$\} \in T$ and for all $\{$`pos,t'`$\}\in T$, `t` is not an initial segment of `t'`.

The QSM algorithm builds a deterministic state machine from a consistent set of traces. The construction fails if applied to an inconsistent set. The machine built, $\mathcal{M}_T =< \mathcal{S}, s_0, \mathcal{F}, \mathcal{T} >$, has the form

- $\mathcal{S}$ is a non-empty finite set of states,
- $s_0 \in \mathcal{S}$ is the initial state of the system,
- $\mathcal{F} \subseteq \mathcal{S}$ is a set of failing states, and
- $\mathcal{T}$ is a set of transitions of the form (s_1, a, s_2) where $s_i \in \mathcal{S}$ and $a \in \mathcal{A}$.

For states s_1 and s_{k+1} and a trace $\mathtt{t} = a_1 a_2 \ldots a_k$ we write $s_1 \xrightarrow{\mathtt{t}} s_{k+1}$ if for each $i = 1 \ldots k$ there is some a_i so that $(s_i, a_i, s_{i+1}) \in \mathcal{T}$.

We say that the machine $\mathcal{M}_T$ *accepts* $\mathtt{t}$ if $s_0 \xrightarrow{\mathtt{t}} s$ with $s \notin \mathcal{F}$; the machine *rejects* $\mathtt{t}$ if $s_0 \xrightarrow{\mathtt{t}} s$ with $s \in \mathcal{F}$, or there is no path through the machine $\mathcal{M}_T$ labelled by $\mathtt{t}$. Since the machine is deterministic, each trace will either be accepted or rejected by the machine, but not both.

The crucial property of the machine $\mathcal{M}_T$ is that for all positive traces $\mathtt{t}$ in T, $\mathcal{M}_T$ accepts $\mathtt{t}$, and for all negative traces $\mathtt{t}$ in T, $\mathcal{M}_T$ rejects $\mathtt{t}$. This is true by construction.

2.2 Abstractions and QSM

We have written $\mathcal{M}_T$ for the machine inferred from a set of traces T, which might be concrete or might arise from an abstraction of a concrete set. In this section we'll work with a fixed set of concrete traces, and use T_α for the set of traces after abstraction and $\mathcal{M}_\alpha$ for the machine generated from T_α.

It would be nice to relate machines produced for related abstractions, e.g. in the case where an abstraction is a composition of two others, with $\alpha = \beta \circ \gamma$. However, even in the case of relating machines for concrete and abstract sets of traces, $\mathcal{M}_{id}$ and $\mathcal{M}_\alpha$, this is not possible in general since the machines generated by the QSM algorithm are not canonical: depending on the order in which states are handled by the algorithm, different machines result.

2.3 SM: state machines

A *state machine* of type $\mathcal{D}$ has the form $\mathcal{M} = < \mathcal{S}, s_0, \mathcal{T}, d_0, \eta, \phi, \psi >$

- $\mathcal{S}$ is a non-empty finite set of states,
- $s_0 \in \mathcal{S}$ is the initial state of the system,

- $\mathcal{T}$ is a set of transitions of the form (s_1, a, s_2) where $s_i \in \mathcal{S}$ and $a \in \mathcal{A}$.
- $d_0 \in \mathcal{D}$ is the initial value of the state data,
- η is a partial function
$$\eta :: \{\mathcal{S}, \mathcal{A}, \mathcal{D}\} \rightarrow \mathcal{D}$$
taking a state, an action and the value of the state data before the transition to the value after,
- ϕ, ψ are partial functions
$$\phi, \psi :: \{\mathcal{S}, \mathcal{A}, \mathcal{D}\} \rightarrow \mathcal{B}$$
where $\mathcal{B}$ is the Boolean type. These are the pre- and post-conditions of the transition from the given state, action and value of the state data.