Illustrating Stepwise Refinement
Shortest Path ASMs

Egon Börger

Dipartimento di Informatica, Università di Pisa
http://www.di.unipi.it/~boerger
For details see Chapter 3.2 (Incremental Design by Refinements) of:

E. Börger, R. Stärk
Abstract State Machines
A Method for High-Level System Design and Analysis
Springer-Verlag 2003
For update info see AsmBook web page:
http://www.di.unipi.it/AsmBook
Shortest Path ASMs: Illustrating Stepwise Refinement

- Computing Graph Reachability Sets: $M_0$
- Wave Propagation of Frontier: $M_1$
- Neighborhoodwise Frontier Propagation: $M_2$
- Edgewise Frontier Extension per Neighborhood: $M_3$
- Queue and Stack Implementation of Frontier and Neighborhoods: $M_4$
- Introducing abstract weights for measuring paths and computing shortest paths: $M_5$ (Moore’s algorithm)
- Instantiating data structures for measures and weights
Computing Graph Reachability Set

• The problem:
  – given a directed graph (NODE, E, source) (here mostly assumed to be finite) with a distinguished source node
  – label every node which is reachable from source via E
  – arrange the labeling so that it terminates for finite graphs

• Solution idea:
  – starting at source, move along edges to neighbor nodes and label every reached node as visited
  – proceed stepwise, pushing in each step the “frontier” of the lastly reached nodes one edge further, without revisiting nodes which have already been labeled as visited
Computing Reachability Set: Machine $M_0$

Initially only source is labeled as visited ($V(\text{source})=1$)

Wave Propagation Rule:

for all $(u,v) \in E$ s.t. $u$ is labeled as visited & $v$ is not labeled as visited
label $v$ as visited

Correctness Lemma:
Each node which is reachable from source is exactly once labeled as visited

Proof. Existence claim: induction on the length of paths from source
Uniqueness property follows from the rule guard enforcing that only nodes not yet labeled as visited are considered for being labeled as visited

Termination Lemma:
For finite graphs, the machine terminates

The meaning of termination:
there is no more edge $(u,v) \in E$ whose tail $u$ is labeled as visited but whose head $v$ is not

Proof. By each rule application, the set of nodes which are not labeled as visited decreases.

© Egon Börger: Shortest Path (ASM Refinements) 5
Identifying the FRONTIER of wave propagation

- frontier = set of nodes lastly labeled as visited (*)
  - Initially: frontier = \{source\} only source is labeled as visited

\[ M_0 \equiv \text{scan} \]
\begin{align*}
\text{forall } u \in \text{frontier} & \quad \text{shift frontier to neighb}(u) \\
\text{delete } u \text{ from frontier}
\end{align*}

\[ M_1 \equiv \text{forall } v \in \text{neighb}
  \begin{align*}
  & \quad \text{shift frontier to } v \\
  & \quad \text{if } v \text{ is not labeled as visited} \\
  & \quad \text{then insert } v \text{ into frontier} \\
  & \quad \text{label } v \text{ as visited}
\end{align*}

\[ \text{neighb}(u) = \{v \mid (u,v) \in E\} \]

\[ \text{label } v \text{ as visited } \equiv \text{visited}(v) := \text{true} \]

NB. Nodes in frontier are labeled as visited

Lemma: \( M_0 / M_1 \) steps are in 1-1 correspondence & perform the same labelings

Proof: by run induction from (*) above
M₁-run computing the reachability set

Frontier propagation: moving frontier simultaneously for each node in frontier to all its neighbors (restricted to those which have not yet been labeled as visited)

In t steps all nodes reachable by a path of length at most t are labeled as visited

Animation courtesy of M. Veanes
Refinement: Shifting frontier to neighborhood of ONE node per step

- determining one next node for frontier propagation by abstract scheduling function select (to be refined later)

\[ M_2 \equiv \]

\[ \text{scan} \]

frontier not empty

\[ \text{let } u = \text{select(frontier)} \text{ in delete } u \text{ from frontier shift frontier to } \text{neighb}(u) \]

Lemma 1. \( \forall t \forall u \in \text{frontier}_t(M_2) \exists t' \leq t \text{ s.t. } u \in \text{frontier}_{t'}(M_1) \)

Proof: Ind(t)

Lemma 2. If \( M_2 \) in step \( t \) labels a node as visited, then \( M_1 \) does the same in some step \( t' \leq t \).

Proof: Ind(t)

Corollary: \( M_1 \) terminates iff \( M_2 \) terminates

Corollary 2: Uniqueness of \( M_1 \)-labeling preserved by \( M_2 \)

Corollary 3: \( M_2 \)-labeling is complete if every node in frontier is eventually selected

assuming finite fan-out
Canonically relating $M_1$- and $M_2$- runs (for finite fan-out)

- Each run of $M_1$ can be simulated by a “breadth-first” run of $M_2$ producing the same labelings of nodes as visited, where each step of $M_1$ applied to frontier ($M_1$) in state $S$ is simulated by selecting successively all the elements of frontier ($M_1$) in state $S$.

$M_1 \equiv \begin{align*}
&\text{scan} \\
&\text{forall } u \in \text{frontier} \\
&\text{delete } u \text{ from frontier} \\
&\text{shift frontier to } \text{neighb}(u)
\end{align*}$

$M_2 \equiv \begin{align*}
\text{scan} \\
\text{if frontier not empty} \\
&\text{let } u = \text{select(frontier)} \\
&\text{delete } u \text{ from frontier} \\
&\text{shift frontier to } \text{neighb}(u)
\end{align*}$
Refinement: Edgewise frontier extension per neighborhood

- Refine $M_2$-rule “shift frontier to neighb(u)” to a submachine \texttt{shift-frontier-to-neighb} which selects one by one every node $v$ of neighb(u) to edgewise “shift frontier to $v$” (using another scheduling fct \texttt{select})

\begin{align*}
\text{shift-frontier-to-neighb (n)} & \equiv \\
\text{initialize neighb by n} & \quad \text{label} \\
\text{neighb not empty} & \quad \text{let v=select(neighb) in} \\
& \quad \text{delete v from neighb} \\
& \quad \text{shift frontier to v}
\end{align*}

- NB. With an appropriate mechanism for the initialization of submachines upon calling, executing $M_2$-rule “shift frontier to neighb(u)” can be replaced by a call to \texttt{shift-frontier-to-neighb(u)}.
Machine with edgewise frontier extension per neighborhood

- Each “shift frontier to neighb(u)” step of $M_2$ is refined by a run of $M_3$-submachine “shift-frontier-to-neighb” with actual parameter neighb(u): started with initializing neighb to neighb(u), iterating “shift frontier to v” for every v in neighb, and exited by returning to scan, thus producing the same labeling of nodes as visited.
- Corollary: Correctness and Termination Lemma carry over from $M_2$ to $M_3$ (assuming finite fan-out and fair scheduling functions)
Refinement of frontier to (fair) queue and of neighb to stack

\[ M_4 \equiv \]

- **Scan**: frontier not empty
- **Label**: let \( u = \text{select}(\text{frontier}) \) in delete \( u \) from frontier initialize \( \text{neighb} \) by \( \text{neighb}(u) \)
- **Neighb**: neighb not empty
- **Shift frontier to \( v \)**

\( \text{frontier as queue: select} = \text{first (at left end)} \quad \text{delete} \ldots \equiv \text{frontier} := \text{rest(frontier)} \)
\( \text{insert} = \text{append (at right end)} \quad \text{NB. No node occurs more than once in frontier} \)

**Neighborhood as stack**
\( \text{select} = \text{top} \quad \text{delete} \equiv \text{pop} \)

for the initialization, \( \text{neighb}(u) \) is assumed to be given as stack for every \( u \)

- Exercise. Prove that \( M_4 \) preserves correctness and termination of \( M_3 \)
- Exercise. Write and test an efficient C++ program for machine \( M_4 \).
Computing the weight of paths from source to determine “shortest” paths to reachable nodes

- Measuring paths by accumulated weight of edges
  - \((M, <)\) well-founded partial order of path measures with
    - smallest element 0 and largest element \(\infty\)
    - greatest lower bound \(\text{glb}(m, m')\) for every \(m, m' \in M\)
  - **edge** weight: \(E \rightarrow \text{WEIGHT}\)
  - \(+: M \times \text{WEIGHT} \rightarrow M\) “adding edge weight to path measure”
    - monotonicity: \(m < m'\) implies \(m + w < m + w\)
    - distributivity wrt \(\text{glb}\): \(\text{glb}(m, m') + w = \text{glb}(m + w, m' + w)\)
  - **path** weight: \(\text{PATH} \rightarrow M\) defined inductively by
    - \(\text{weight}(\varepsilon) = 0\)
    - \(\text{weight}(pe) = \text{weight}(p) + \text{weight}(e)\)
Computing minimal weight of paths

- **min-weight**: \( \text{NODE} \rightarrow \text{M} \) defined by
  - \( \text{min-weight}(u) = \text{glb}\{\text{weight}(p) | p \text{ is a path from source to } u\} \)

- **NB.** The function is well-defined since by the well-foundedness of \(<\), countable sets of measures (which may occur due to paths with cycles) have a glb

- Successive approximation of **min-weight from above** for nodes reachable from source by a function

  **up-bd**: \( \text{NODE} \rightarrow \text{M} \)
  - initially \( \text{up-bd}(u) = \infty \) for all \( u \) except \( \text{up-bd}(\text{source}) = 0 \)
  - for every \( v \) reachable by an edge \( e \) from \( u \) s.t. \( \text{up-bd}(v) \) can be decreased via \( \text{up-bd}(u) + \text{weight}(e) \),
    - **lower up-bd** \( v \) to \( \text{glb}\{\text{up-bd}(v), \text{up-bd}(u) + \text{weight}(e)\} \)

- **NB.** If not \( \text{up-bd}(v) \leq \text{up-bd}(u) + \text{weight}(e) \), then
  - \( \text{glb}\{\text{up-bd}(v), \text{up-bd}(u) + \text{weight}(e)\} < \text{up-bd}(v) \)
Refining $M_4$ to compute $\text{up-bd} \geq \text{min-weight}$:

**same machine** refining “frontier shift” to “lowering up-bd”

- **Initially**: frontier = \{source\}  \text{ctl-state} = \text{scan}
- $\text{up-bd}(u) = \infty$ for all $u$ except $\text{up-bd}(\text{source}) = 0$

### Algorithm

- **frontier not empty**
  - let $u = \text{select}($frontier$)$ in
  - delete $u$ from frontier
  - initialize neighb by $(u,\text{neighb}(u))$

- **neighb not empty**
  - let $v = \text{select}($neighb$)$ in
  - delete $v$ from neighb
  - shift frontier to $v$

- **lower up-bd(v) via u**
  - if not $\text{up-bd}(v) \leq \text{up-bd}(u) + \text{weight}(u,v)$ then
  - $\text{up-bd}(v) := \text{glb}\{\text{up-bd}(v), \text{up-bd}(u) + \text{weight}(u,v)\}$
  - if $v \notin$ frontier then insert $v$ into frontier
Moore’s algorithm $M_5$ terminates (for finite graphs)
- each scan step diminishes the size of frontier
- each label step shrinks neighb; each head node $v$ upon entering frontier gets $\text{up-bd}(v)$ updated to a smaller value. Since $<$ is well-founded, this can happen only finitely often.
Correctness Proof for the computation of min-weight

• **Theorem.** When Moore’s algorithm $M_5$ terminates, $\minweight(u) = \uppbd(u)$ for every $u$.
  – **Proof.** $\minweight(u) \leq \uppbd(u)$ (lemma 1). Since $\uppbd(u)$ is a lower bound for $\weight(p)$ for every path $p$ from source to $u$ (lemma 2) and since $\minweight$ by definition is the glb of such path weights, also $\geq$ holds.

• **Lemma 1.** At each step $t$ and for each $v$: $\minweight(v) \leq \uppbd(v)_t$.

• **Lemma 2.** When $M_5$ terminates, $\uppbd(v) \leq \weight(p)$ for every path $p$ from source to $v$. 

© Egon Börger: Shortest Path (ASM Refinements) 17
Proof for the approximation of min-weight by up-bd

• **Lemma 1.** At each step $t$, for each $v$: $\text{min-weight}(v) \leq \text{up-bd}(v)_t$.
  
  – **Proof 1.** Ind$(t)$. For $t=0$ the claim holds by definition.

• At $t+1$ (only) rule “lower up-bd(v) via u” sets $\text{up-bd}(v)_{t+1}$, namely to $\text{glb}\{\text{up-bd}(v)_t, \text{up-bd}(u)_t + \text{weight}(u,v)\}$. Remains to show
  
  – $\text{min-weight}(v) \leq \text{up-bd}(v)_t$ (which is true by ind.hyp. for $v$)
  
  – $\text{min-weight}(v) \leq \text{up-bd}(u)_t + \text{weight}(u,v)$

• The latter relation follows from

  (*) $\text{min-weight}(v) \leq \text{min-weight}(u) + \text{weight}(u,v)$

  by $\text{min-weight}(u) \leq \text{up-bd}(u)_t$ (ind.hyp.) via monotonicity of $+$

• *ad (*)*: $\text{glb}\{\text{weight}(p) | p \text{ path from source to } v\} \leq \text{glb}\{\text{weight}(p.(u,v)) | p \text{ path from source to } u\}$ = def $\text{weight}$

  $\text{glb}\{\text{weight}(p) + \text{weight}(u,v) | p \text{ path from source to } u\}$ = glb distrib

  $\text{glb}\{\text{weight}(p) | p \text{ path from source to } u\} + \text{weight}(u,v)$

  = $\text{min-weight} \text{ min-weight}(u) + \text{weight}(u,v)$

**lower up-bd(v) via u** $\equiv$ if not $\text{up-bd}(v) \leq \text{up-bd}(u) + \text{weight}(u,v)$ then

$\text{up-bd}(v):= \text{glb}\{\text{up-bd}(v), \text{up-bd}(u) + \text{weight}(u,v)\}$

if $v \notin \text{frontier}$ then insert $v$ into $\text{frontier}$
Proof for lower bound $\text{up-bd}(v)$ of weight of paths to $v$

- **Lemma 2.** When $M_5$ terminates, $\text{up-bd}(v) \leq \text{weight}(p)$ for every path $p$ from source to $v$.
  - **Proof 2.** Ind(path length). For $t=0$ the claim holds by definition.

- Let $p.(u,v)$ be a path of length $t+1$.

- $\text{up-bd}(v) \leq \text{up-bd}(u) + \text{weight}(u,v)$

  - by termination of $M_5$ (otherwise *lower up-bd*(v) via u could fire)

- $\text{up-bd}(u) \leq \text{weight}(p)$ (ind.hyp.), thus by monotonicity of +

  $\text{up-bd}(u) + \text{weight}(u,v) \leq \text{weight}(p) + \text{weight}(u,v)$

  $=_{\text{def weight}} \text{weight}(p.(u,v))$

---

**lower up-bd(v) via u**  
≡ if not $\text{up-bd}(v) \leq \text{up-bd}(u)+\text{weight}(u,v)$ then

$\text{up-bd}(v) := \text{glb}\{\text{up-bd}(v), \text{up-bd}(u)+\text{weight}(u,v)\}$

if $v \notin \text{frontier}$ then insert $v$ into frontier
Instantiating data structures for weight and measure

• \((M,\prec) = (\text{Nat} \cup \{\infty\},\prec)\) well-founded order of shortest path measures with
  • smallest element 0 and largest element \(\infty\)
  • greatest lower bound \(\text{glb}(m,m') = \min(m,m')\)
• \(\text{WEIGHT} = (\text{Nat}, +)\) with \(n + \infty = \infty\)
  • monotonicity: \(m < m'\) implies \(m + w < m' + w\)
  • \(\text{glb}\) distributive wrt \(+\): \(\text{glb}(m + w, m' + w) = \text{glb}(m, m') + w\)
• For an instantiation to the constrained shortest path problem see K. Stroetmann’s paper in JUCS 1997.
• For Dijkstra’s refinement \(M_5\) see Ch.3.2.1 of the AsmBook
References

• E. F. Moore: The Shortest Path Through a Maze.

• K. Stroetmann: The Constrained Shortest Path Problem: A Case Study in Using ASMs

• E. Börger, R. Stärk: Abstract State Machines. A Method for High-Level System Design and Analysis Springer-Verlag 2003, see Chapter 3.2.1
  http://www.di.unipi.it/AsmBook