Illustrating Stepwise Refinement
Shortest Path ASMs

Egon Börger

Dipartimento di Informatica, Universita di Pisa
http://www.di.unipi.it/~boerger

Shortest Path ASMs: Illustrating Stepwise Refinement

• Computing Graph Reachability Sets: M₀
• Wave Propagation of Frontier: M₁
• Neighborhoodwise Frontier Propagation: M₂
• Edgewise Frontier Extension per Neighborhood: M₃
• Queue and Stack Implementation of Frontier and Neighborhoods: M₄
• Introducing abstract weights for measuring paths and computing shortest paths: M₅ (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 last reached nodes one edge further, without revisiting nodes which have already been labeled as visited

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
Computing Reachability Set: Machine $M_0$

**Initial**y 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

**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 ensuring that only nodes not yet labeled as visited are considered for being labeled as visited.

**Termination Lemma:**

For finite graphs, the machine terminates.

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

**M$_1$-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).

**M$_2$-run**

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)

**Refinement:**

Shifting frontier to neighborhood of ONE node per step.

- identifying the FRONTIER of wave propagation

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

**M$_1$**

- \(\text{scan}\)
- shift frontier to neighb($u$)
- delete $u$ from frontier

**M$_2$**

- \(\text{scan}\)
- let $u=\text{select}(\text{frontier})$ in
- delete $u$ from frontier
- shift frontier to neighb($u$)

**Lemma:** $M_0 / M_1$ steps are in 1-1 correspondence & perform the same labelings.

**Proof:** by run induction from (*) above
Can 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 $S$.

$M_1 \equiv$

$M_2 \equiv$

Refinement: Edgewise frontier extension per neighborhood

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

$\text{shift-frontier-to-neighb} (n) \equiv$

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.

$M_3 \equiv$

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

$M_4 \equiv$

Exercise. Prove that $M_4$ preserves correctness and termination of $M_3$.

Exercise. Write and test an efficient C++ program for machine $M_4$.  

frontier as queue: select = first (at left end) delete … = frontier := rest(frontier)

neighborhood as stack select = top delete $\equiv$ pop

insert = append (at right end) NB. No node occurs more than once in frontier

for the initialization, neighb(u) is assumed to be given as stack for every $u$. 

© Egon Börger: Shortest Path (ASM Refinements)
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 \(m, m' \in M\)
- \(\text{edge weight: } E \to \text{WEIGHT} \)
- \(+: M \times \text{WEIGHT} \to M\) “adding edge weight to path measure”
  - monotonicity: \(m < m'\) implies \(m + w < m + w\)
  - distributivity wrt \(\text{glb}: \text{glb}(m, m') + w = \text{glb}(m + w, m' + w)\)
- \(\text{path weight: } \text{PATH} \to M\) defined inductively by
  - \(\text{weight(}\varepsilon\text{)} = 0\)
  - \(\text{weight}(pe) = \text{weight}(p) + \text{weight}(e)\)

Refining \(M_4\) to compute \(\text{up-bd} \geq \text{min-weight}\):
- Initially: \(\text{frontier} = \{\text{source}\} \quad \text{ctl-state} = \text{scan}\)
- \(\text{up-bd}(u) = \infty\) for all \(u\) except \(\text{up-bd}(\text{source}) = 0\)

Refining termination and completeness proofs for \(M_5\)
- Moore’s algorithm \(M_5\) terminates (for finite graphs)
  - each scan step diminishes the size of frontier
  - each label step shrinks \(\text{neighb}\); each head node \(v\) upon entering \(\text{frontier}\) 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, $\text{min-weight}(u) = \text{up-bd}(u)$ for every $u$.
  – Proof. $\text{min-weight}(u) \leq \text{up-bd}(u)$ (lemma 1). Since $\text{up-bd}(u)$ is a lower bound for $\text{weight}(p)$ for every path $p$ from source to $u$ (lemma 2) and since $\text{min-weight}$ by definition is the glb of such path weights, also $\geq$ holds.

• Lemma 1. At each step $t$ and for each $v$: $\text{min-weight}(v) \leq \text{up-bd}(v)_t$.

• Lemma 2. When $M_5$ terminates, $\text{up-bd}(v) \leq \text{weight}(p)$ for every path $p$ from source to $v$.

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 $\text{up-bd}(v)$ via $u$ could fire)
  • $\text{up-bd}(u) \leq \text{weight}(p)$ (ind.hyp.), thus by monotonicity of $+$
  • $\text{up-bd}(u) + \text{weight}(u,v) \leq \text{weight}(p) + \text{weight}(u,v)$
    = $\text{def weight}$ $\text{weight}(p.(u,v))$

Instantiating data structures for weight and measure

• $(M, <) = (\mathbb{N} \cup \{\infty\}, <)$ well-founded order of shortest path measures with
  • smallest element 0 and largest element $\infty$
  • greatest lower bound $\text{glb}(m, m') = \text{min}(m, m')$

• $\text{WEIGHT} = (\mathbb{N}, +)$ 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.
  – Proc. International Symposium on the Theory of Switching,
    Laboratory of Harvard University”, Cambridge, MA, 1959,
    Harvard University Press.

• 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