Illustrating Stepwise Refinement
Shortest Path ASMs

Egon Börger

Dipartimento di Informatica, Universita 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(source)=1)

Wave Propagation Rule:

for all (u,v) Î 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

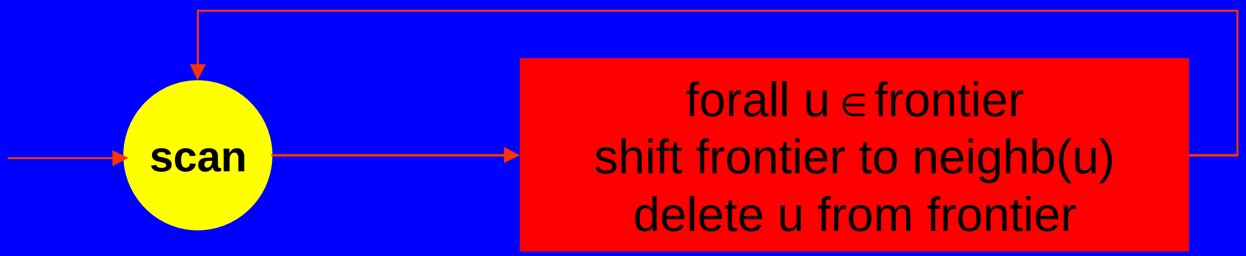
Proof. By each rule application, the set of nodes which are not labeled as visited decreases.

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

Identifying the FRONTIER of wave propagation

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

$M_1 \equiv$



shift frontier to neighb ≡
forall v ∈ neighb
shift frontier to v

neigb(u) = {v| (u,v) ∈ E}

label v as visited ≡
visited(v):= true

shift frontier to v ≡ if v is not labeled as visited
then insert v into frontier
label v as visited

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

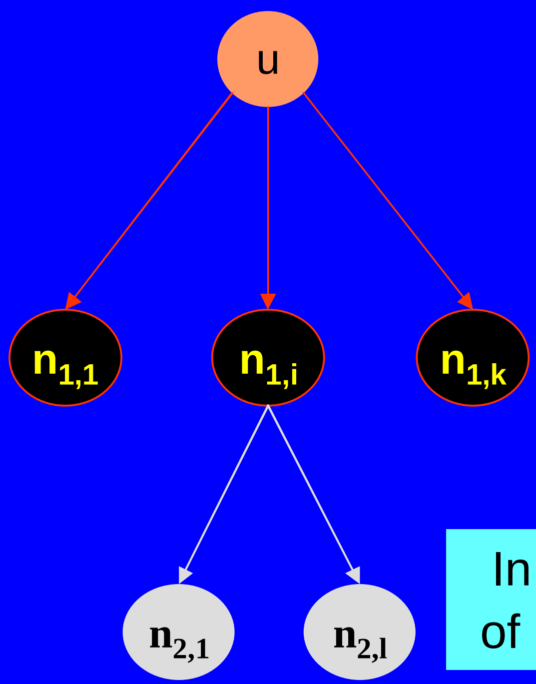
M_1-run computing the reachability set

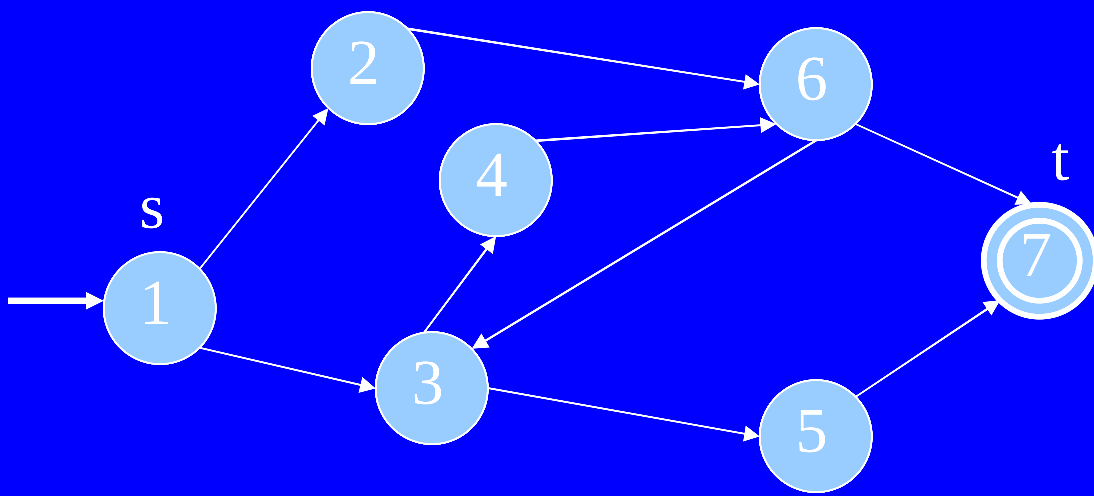
Frontier propagation: moving frontier simultaneously for each node in frontier to all its neighbors (restricted to those which have not yet been labeled as visited)

First step

visited

frontier





neighb(u)

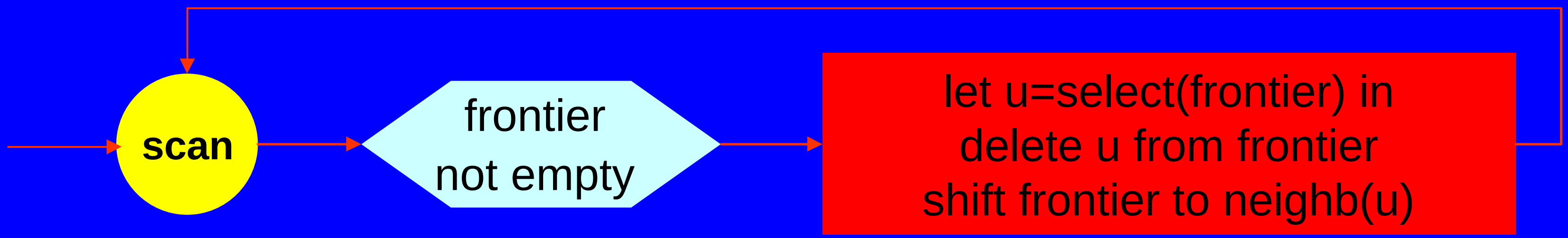
In t steps all nodes reachable by a path of length at most t are labeled as visited

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)

$M_2 \equiv$



Lemma 1. $\forall t \, \forall u \in frontier_t(M_2) \, \exists t' \leq t$ s.t. $u \in frontier_{t'}(M_1)$

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)

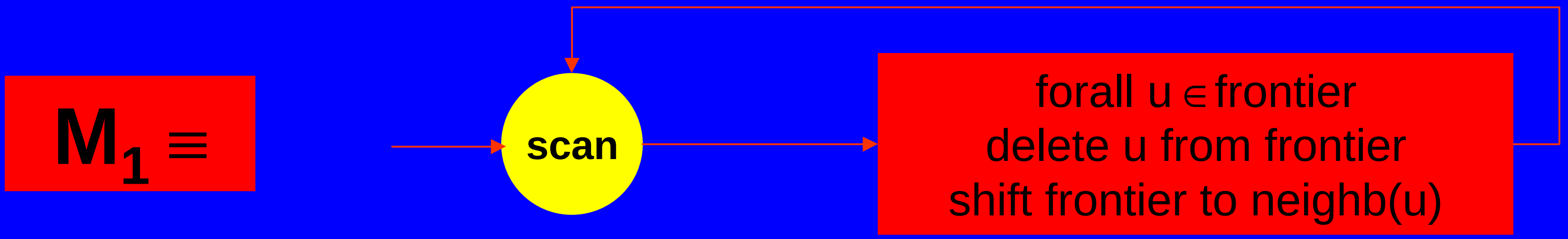
Corollary: M_1 terminates iff M_2 terminates

Corollary 2: Uniqueness of M_1-labeling preserved by M_2

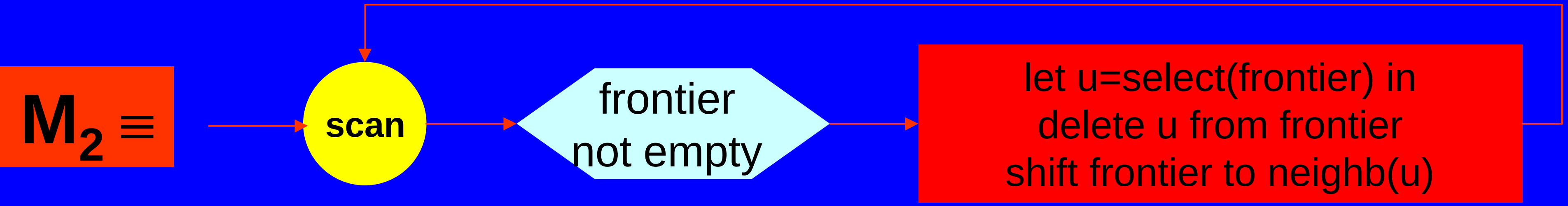
assuming finite fan-out

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

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



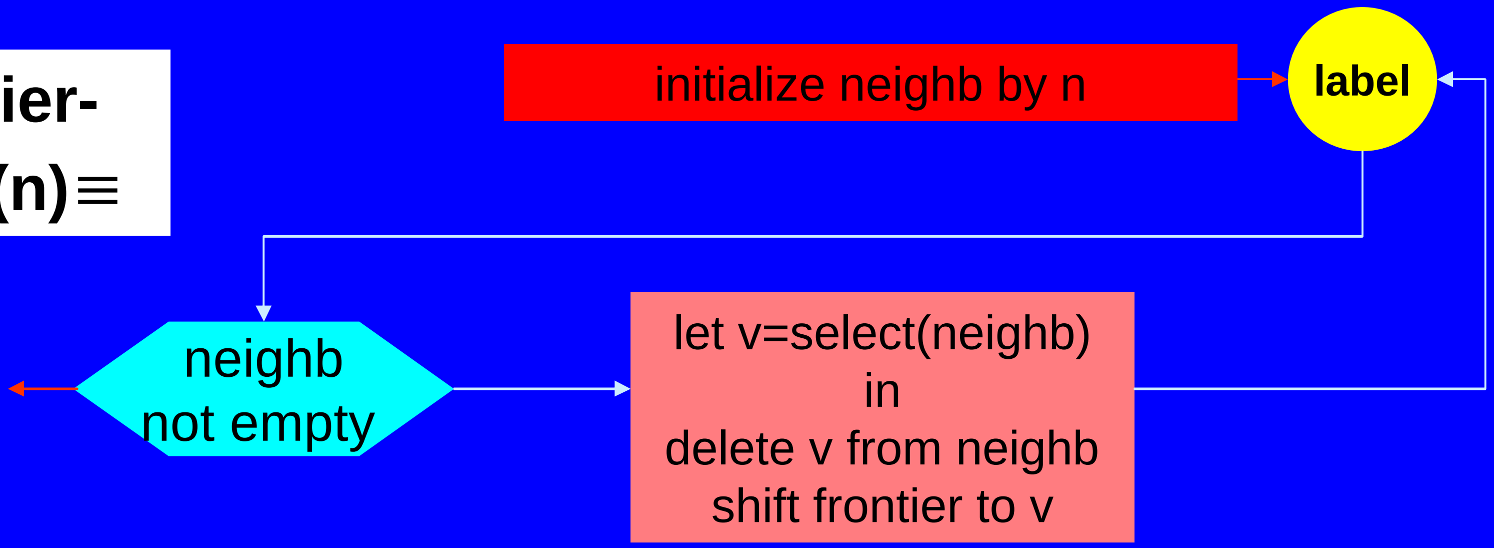
- Each run of M_1 can be simulated by a “breadth-first” run of M_2 producing the same labelings of nodes as visited, where each step of M_1 applied to frontier (M_1) in state S is simulated by selecting successively all the elements of frontier (M_1) in state S.



Refinement: Edgewise frontier extension per neighborhood

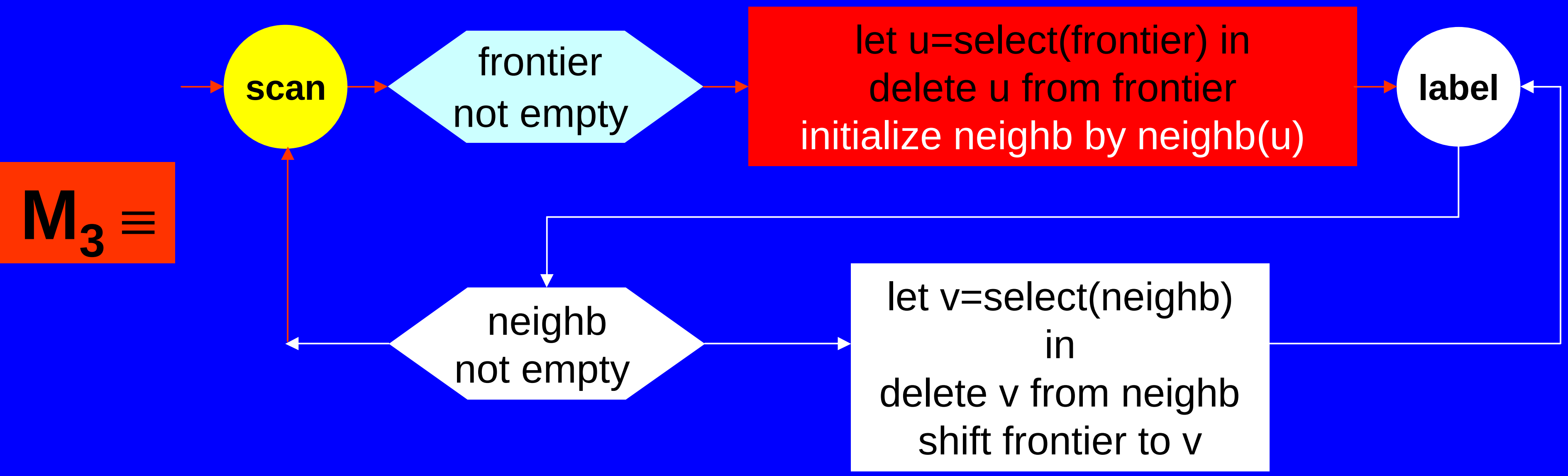
- Refine M_2-rule “shift frontier to neighb(u)” to a submachine shift-frontier-to-neighb which selects one by one every node v of neighb(u) to edgewise “shift frontier to v” (using another scheduling fct select)

shift-frontier-to-neighb (n)≡



- 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 shift-frontier-to-neighb(u).

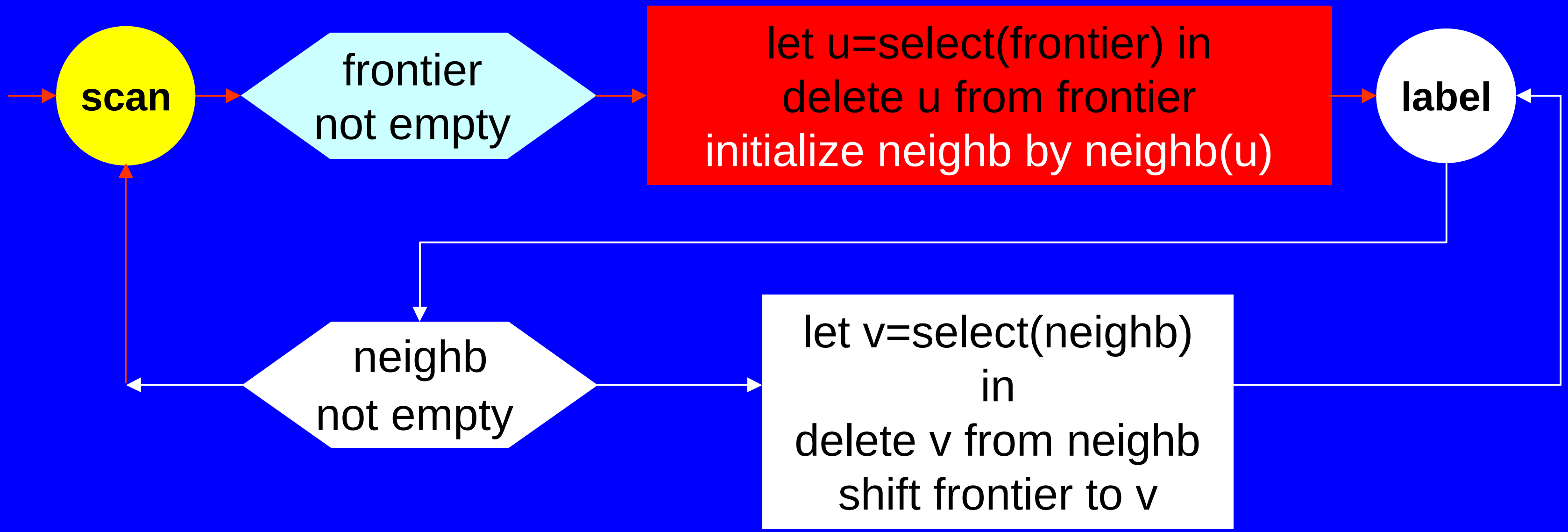
Machine with edgewise frontier extension per neighborhood



- 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

$M_4 \equiv$



frontier as queue: select = first (at left end) delete … ≡ frontier := rest(frontier)
insert = append (at right end) NB. No node occurs more than once in frontier

neighborhood as stack select = top delete ≡ pop
for the initialization, 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 ∞
    - greatest lower bound glb(m,m’) for every m,m’$\in$M
  - edge weight: E $\rightarrow$ WEIGHT
  - +: M $\times$ WEIGHT $\rightarrow$ M “adding edge weight to path measure”
    - monotonicity: m < m’ implies m + w < m + w
    - distributivity wrt glb: glb(m,m’) + w = glb(m + w,m’ + w)
  - path weight:PATH $\rightarrow$ M defined inductively by
    - weight(ε) = 0
    - weight(pe)= weight(p)+weight(e)

Computing minimal weight of paths

- min-weight: NODE $\rightarrow$ M defined by
  – min-weight(u) = glb{weight(p)| p 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: NODE $\rightarrow$ M

  – initially up-bd(u) = ∞ for all u except up-bd(source) = 0
  – for every v reachable by an edge e from u s.t. up-bd(v) can be decreased via up-bd(u)+weight(e),

  lower up-bd(v) to glb{up-bd(v), up-bd(u)+weight(e)}

  - NB. If not up-bd(v) $\leq$ up-bd(u)+weight(e), then glb{up-bd(v), up-bd(u)+weight(e)} $<$ up-bd(v)

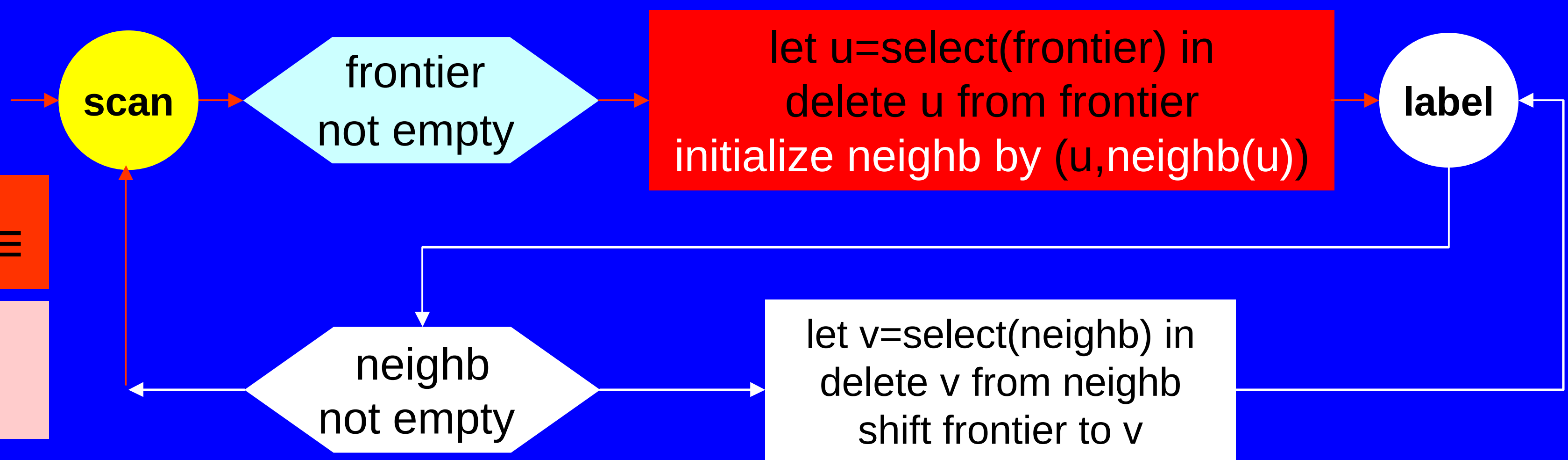
Refining M_4 to compute up-bd ≥min-weight:

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

- Initially: frontier = {source}   ctl-state = scan
- up-bd(u)= ∞ for all u except up-bd(source) = 0

$M_{4/5}$ ≡

Moore 1959



shift frontier to v ≡
if v is not labeled as visited then
label v as visited
insert v into frontier

NB.frontier not a multi-set

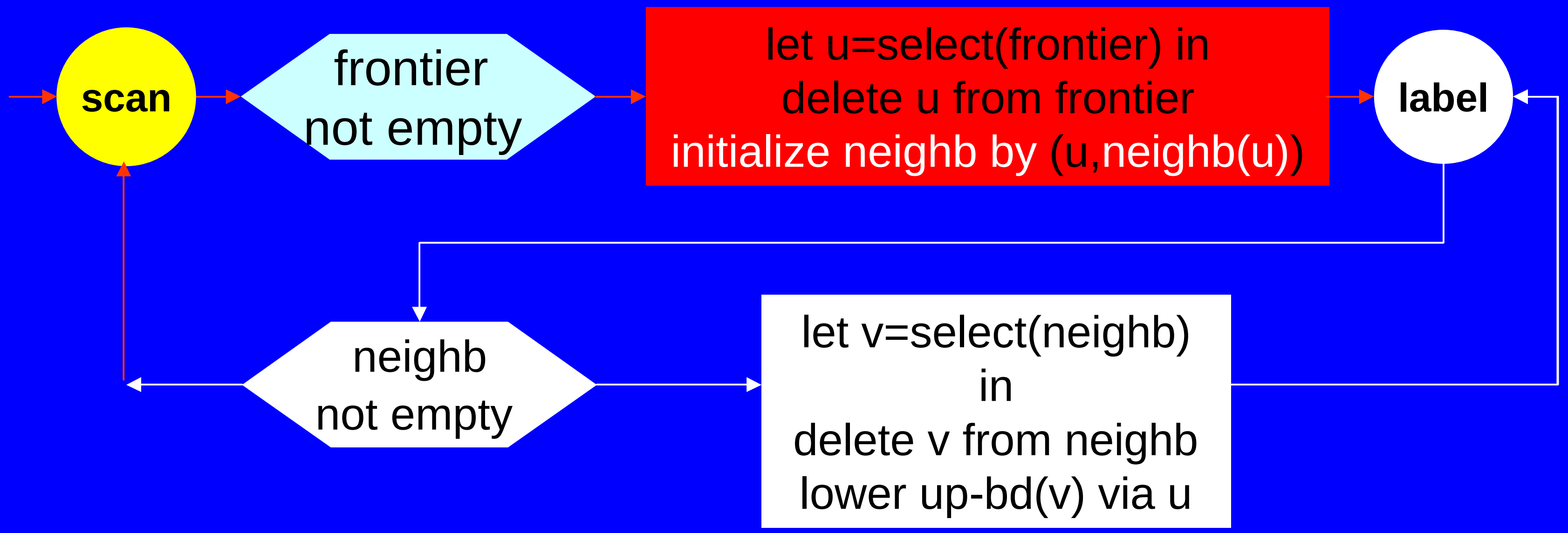
lower up-bd(v) via u

shift frontier to v ≡

if not up-bd(v) ≤ up-bd(u)+weight(u,v) then
up-bd(v):= glb{up-bd(v), up-bd(u)+weight(u,v)}
if v∉frontier then insert v into frontier

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 neighb; each head node v upon entering frontier gets 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, min-weight(u)= up-bd(u) for every u.
  - Proof. min-weight(u) ≤up-bd(u) (lemma 1). Since up-bd(u) is a lower bound for weight(p) for every path p from source to u (lemma 2) and since min-weight by definition is the glb of such path weights, also ≥ holds.
- Lemma 1. At each step t and for each v: min-weight(v) ≤$\text{up-bd}(v)_t$.
- Lemma 2. When M_5 terminates, up-bd(v) ≤ weight(p) for every path p from source to v.

Proof for the approximation of min-weight by up-bd

- Lemma 1. At each step t, for each v: min-weight(v) ≤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 up-bd(v)$_{t+1}$, namely to glb{up-bd(v)$_t$, up-bd(u)$_t$ +weight(u,v)}. Remains to show
  - min-weight(v) ≤up-bd(v)$_t$ (which is true by ind.hyp. for v)
  - min-weight(v) ≤ up-bd(u)$_t$ +weight(u,v)
- The latter relation follows from

(*) min-weight(v) ≤ min-weight(u)+weight(u,v)

by min-weight(u) ≤up-bd(u)$_t$ (ind.hyp.) via monotonicity of +

- ad (*): glb({weight(p)| p path from source to v}) ≤
  glb({weight(p.(u,v)) | p path from source to u}) =$_{\text{def weight}}$
  glb({weight(p)+weight(u,v) | p path from source to u}) =$_{\text{glb distrib}}$
  glb({weight(p) | p path from source to u}) +weight(u,v)
  =$_{\text{min-weight}}$ min-weight(u) +weight(u,v)

lower up-bd(v) via u ≡ if not up-bd(v) ≤ up-bd(u)+weight(u,v) then
up-bd(v):= glb{up-bd(v), up-bd(u)+weight(u,v)}
if v∉ frontier then insert v into frontier

Proof for lower bound up-bd(v) of weight of paths to v

- Lemma 2. When M_5 terminates, up-bd(v) ≤ 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.
- up-bd(v) ≤ up-bd(u) +weight(u,v)
  - by termination of M_5 (otherwise **lower up-bd(v) via u** could fire)
- up-bd(u) ≤ weight(p) (ind.hyp.), thus by monotonicity of +

up-bd(u) +weight(u,v) ≤ weight(p) +weight(u,v)

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

lower up-bd(v) via u ≡ if not up-bd(v) ≤ up-bd(u)+weight(u,v) then
up-bd(v):= glb{up-bd(v), up-bd(u)+weight(u,v)}
if v∉ frontier then insert v into frontier

Instantiating data structures for weight and measure

- $(M,<) = (Nat \cup \{\infty\},<)$ well-founded order of shortest path measures with
  - smallest element 0 and largest element ∞
  - greatest lower bound glb(m,m') = min(m,m')
- WEIGHT = (Nat, +) with $n+\infty=\infty$
  - monotonicity: m<m' implies m+w<m'+w
  - glb distributive wrt +: glb(m +w,m' +w) = 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, Part II, Vol. 30 of “The Annals of the Computation Laboratory of Harvard University”, Cambridge, MA, 1959, Harvard University Press.
- K. Stroetmann: The Constrained Shortest Path Problem: A Case Study in Using ASMs
  - In: J. of Universal Computer Science 3 (4), 1997.
- 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