MODEL-CHECKING IN DENSE REAL-TIME

SHANT HARUTUNIAN

1. Introduction

These slides are for a talk based on the paper **Model-Checking in Dense Real-Time**, by Rajeev Alur, Costas Courcoubetis, and David Dill. The paper was published in *Information and Computation* 104(1):2-34, 1993 (preliminary version appeared in *Proc. 5th LICS*, 1990).
A URL to the paper is http://www.cis.upenn.edu/ alur/Lics90D.ps.gz.

The overview of CTL is based on a book chapter titled **Model Checking and the Mu-calculus** by E. Allen Emerson. This was published in *Proceedings of the DIMACS Symposium on Descriptive Complexity and Finite Model*, N. Immerman and P. Kolaitis, eds., American Mathematical Society Press, Pages 185-214. A URL to the book chapter is http://www.cs.utexas.edu/users/emerson/pubs/fmt96q.ps.

2. CTL (Computation Tree Logic)

2.1. **Kripke Structure.** A Kripke Structure is a triple (S, L, R), where

- S is a set of states.
- L is a mapping $L : S \to 2^{AP}$, where AP is a set of atomic propositions.
- $R \subseteq S \times S$ is a total relation, $\forall_{s \in S}\ \exists_{t \in S}$ s.t. $(s, t) \in R$

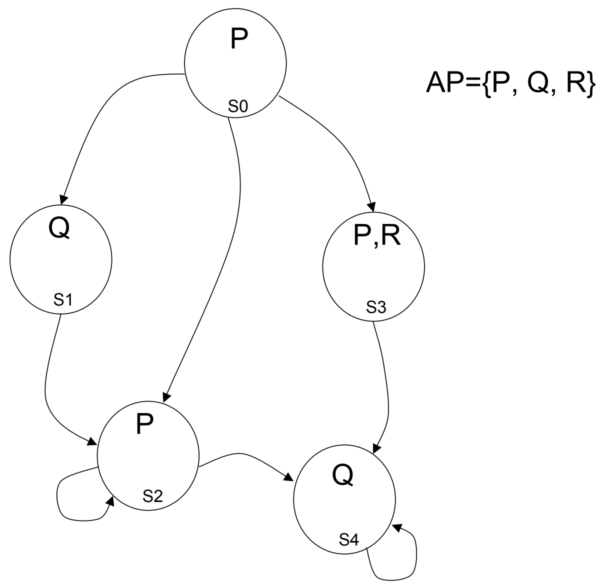

FIGURE 1. Sample Kripke Structure

2.2. **Syntax.** CTL is inductively defined as follows

- S1 A proposition p in AP is a state formula.
- S2 If p and q are state formula, then $p \wedge q$, $\neg p$ are a state formula.
- S3 If p is a path formula, then Ep and Ap are state formula.
- P0 If p and q are state formula, then Xp and pUq are path formula.

The state formulas generated by S1-S3 define the language of CTL.

Alternative rules, (replace S3 and P0 with Sa below).

Sa If p and q are state formula, then AXp, EXp, $ApUq$, and $EpUq$ are state formula.

We use the following abbreviations:

- EFp for $E\ true\ Up$
- AFp for $A\ true\ Up$
- EGp for $\neg(A\ true\ U\neg p)$
- AGp for $\neg(E\ true\ U\neg p)$

Some sample CTL formulas are as follows:

- EXp
- $ApUq$
- $AG(p \Rightarrow AFq)$

2.3. **Full Path.**

- A full path is an infinite sequence of states
  s_0, s_1, s_2, ..., where $(s_i, s_{i+1}) \in R$
- For a full path $x = (s_0, s_1, s_2, \ldots)$,
  we denote by $x^i = (s_i, s_{i+1}, s_{i+2}, \ldots)$.

2.4. **CTL Semantics.**

- For a Kripke structure M and a state s_0, we write $M, s_0 \models p$, for a state formula p

- For a Kripke structure M and a full path x, we write $M, x \models p$, for a path formula p

We define $\models$ inductively:

S1 $M, s_0 \models p$ iff $p \in L(s_0)$, for $p \in AP$

S2 $M, s_0 \models p \wedge q$ iff $M, s_0 \models p$ and $M, s_0 \models q$
$M, s_0 \models \neg p$ iff it is not the case that $M, s_0 \models p$

S3 $M, s_0 \models Ep$ iff $\exists$ a full path $x = (s_0, s_1, s_2, \ldots)$ in M, and $M, x \models p$
$M, s_0 \models Ap$ iff $\forall$ full paths $x = (s_0, s_1, s_2, \ldots)$ in M, and $M, x \models p$

P0 $M, x \models pUq$ iff $\exists i$, $M, s_i \models q$ and $\forall_{j<i}$, $M, s_j \models p$
$M, x \models Xp$ iff $M, s_1 \models p$

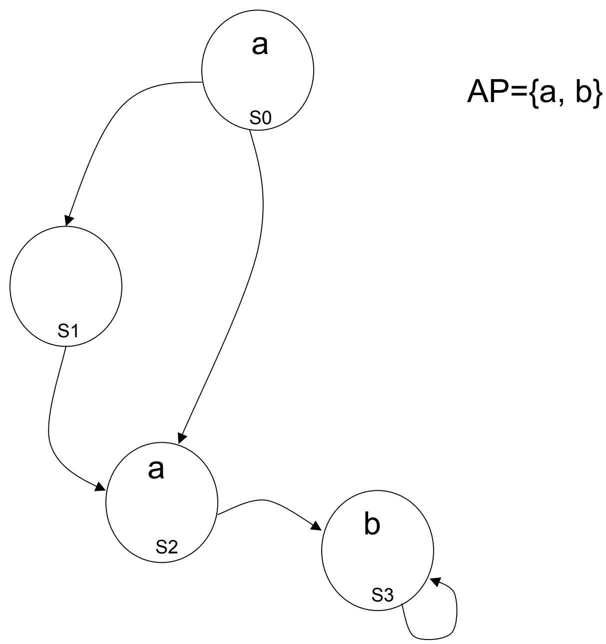

FIGURE 2. Example CTL Model-Checking

We wish to determine for which states of the Kripke structure the property $\phi = EaUb$ holds.

We use the following algorithm to Model-Check the formula ϕ=$EaUb$.

1 `let` D `=` $\emptyset$

2 `for all` $s \in S$`, if` $b \in L(s)$`, then`

    `add` s `to` D`, and`

    `let` $L(s) = L(s) \cup \{EaUb\}$

3 H `=` $\emptyset$

4 `While` $H \neq D$ `do`

  4.1 $H = D$

  4.2 `for all` $s \in S \backslash H$`,`

      `if` $\exists_t$ $(s,t) \in R$`, and` $t \in H$`, and` $a \in L(s)$`,`

      `then`

        `add` s `to` D`, and`

        `let` $L(s) = L(s) \cup \{EaUb\}$

5 `od`

We step through the algorithm for the example structure.

2 $D = \{s_3\}$, and $L(s_3) = \{b\} \cup \{EaUb\}$

3 $H = \emptyset$

4.1,i1 $H = \{s_3\}$

4.2,i1 $S \backslash H = \{s_0, s_1, s_2\}$

$D = \{s_3, s_2\}$, `(we add` s_2 `to the set)`

4.3,i1 $L(s_2) = \{a\} \cup \{EaUb\}$,`(we add` ϕ `to the labels of` s_2`)`

4.1,i2 $H = \{s_3, s_2\}$

4.2,i2 $S \backslash H = \{s_0, s_1\}$

$D = \{s_3, s_2, s_0\}$, `(we add` s_0 `to the set)`

4.3,i2 $L(s_0) = \{a\} \cup \{EaUb\}$,`(we add` ϕ `to the labels of` s_0`)`

4.1,i3 $H = \{s_3, s_2, s_0\}$

4.2,i3 $S \backslash H = \{s_1\}$

$D = \{s_3, s_2, s_0\}$, `(nothing is added to the set)`

5 Exit loop `(we exit the loop since` $H = D$`)`

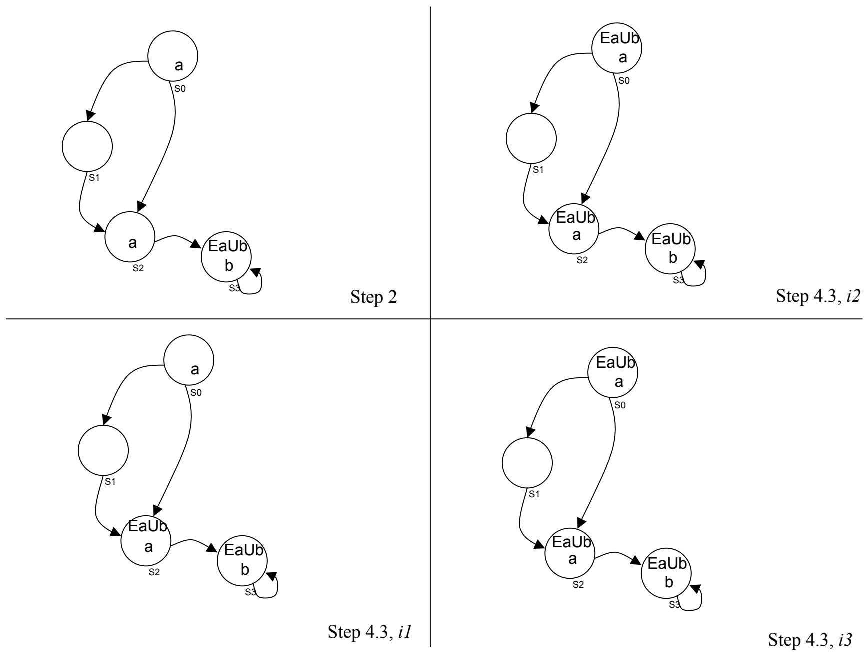

FIGURE 3. Labelled Kripke Structure at various steps in the Model-Checking Algorithm

3. Model-Checking in Dense Real-Time

3.1. Timed Graph. A tuple $(S, \mu, S_{init}, E, C, \pi, \tau)$

- S: A finite set of *nodes*.
- S_{init}: A node in S designated as the start node.
- μ: $S \to 2^{AP}$, where AP is a set of atomic propositions.
- E: $E \subseteq S \times S$, the set of edges.
- C: Finite set of clocks
  - A clock is a variable ranging over the nonnegative Reals

π: $E \rightarrow 2^C$, indicates which clocks in C are reset along an edge in E.

τ: A function labelling each edge in E with an enabling condition built from boolean connectives of atomic formula of the form

$X \leq c$

$c \leq X$

where X is a clock and $c \in N$.

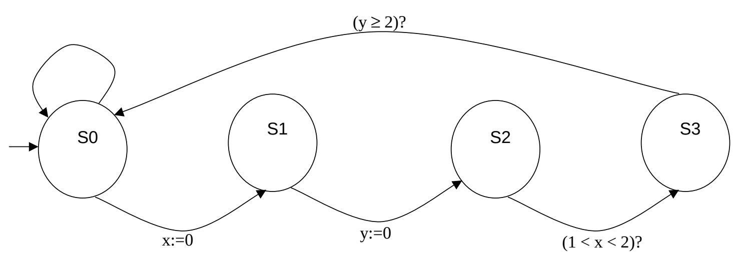

FIGURE 4. Sample Timed Graph

3.2. Clock Assignments.

A clock assignments ν assigns a nonnegative real value to each clock in C, $\nu : C \to R$.

We let $\Gamma(G)$ denote the set of clock assignments for a timed graph G.

We use the following notation regarding clock assignments:

$\nu + t$ for each $y \in C$, $[\nu + t](y) = \nu(y) + t$

$[x \mapsto t]\nu$: for each $y \in C$

$y \neq x, [x \mapsto t]\nu(y) = \nu(y)$

$y = x, [x \mapsto t]\nu(y) = t$

3.3. (s, ν)-Run of a timed graph.

An *infinite* sequence of the following form

$(\langle s_0, \nu_0, t_0 \rangle,\ \langle s_1, \nu_1, t_1 \rangle,\ \langle s_2, \nu_2, t_2 \rangle,\ \ldots)$

Initialization: $s_0 = s$, $\nu_0 = \nu$, and $t_0 = 0$.

Consecution: We have the following requirements regarding a transition from one component of the run to the next:

$t_{i+1} > t_i$.

For edge $e_i \in E$, $e_i = \langle s_i, s_{i+1} \rangle$.

$\nu_{i+1}{=}[\pi(e_i) \mapsto 0](\nu_i + t_{i+1} - t_i)$.

$(\nu_i{+}t_{i+1}{-}t_i)$ satisfies the enabling condition, $\tau(e_i)$.

Progress of time: For any $t \in R$, there exists i s.t. $t_i \geq t$.

3.4. (s, ν)-Path.

We may derive a (s, ν)-Path from a (s, ν)-Run

$$\rho : R \to S \times \Gamma(G)$$

$$\rho(t) = \langle s_j, \nu_j + t - t_j \rangle \text{ for } t_j \leq t < t_{j+1}$$

3.5. Example (s, ν)-Run of a timed graph.

r_1 $(\langle s_0, [0, 0], 0\rangle,$ `(where` $[0,0]$ `is` $[\nu_0(x), \nu_0(y)]$`)`

$\langle s_1, [0, 0.5], 0.5\rangle,$

$\langle s_2, [1, 0], 1.5\rangle,$

$\langle s_3, [1.7, 0.7], 2.2\rangle,$

$\langle s_0, [3.7, 2.7], 4.2\rangle,$

$\langle s_1, [0, 2.8], 4.3\rangle,$

$\langle s_2, [0.1, 0], 4.4\rangle,$

$\langle s_3, [1.1, 1], 5.4\rangle,$

$\langle s_0, [3.1, 3], 4.2 + 3.2i\rangle,$

$\langle s_1, [0, 3.1], 4.2 + 3.2i + 0.1\rangle,$

$\langle s_2, [0.1, 0], 4.2 + 3.2i + 0.2\rangle,$

$\langle s_3, [1.1, 1], 4.2 + 3.2i + 1.2\rangle)$, for all $i > 0$.

ρ_{r_1}: $\rho_{r_1}(4.25) = \langle s_0, [3.75, 2.75]\rangle$

3.6. **Example Sequences that are NOT Runs.**

seq_1

$$(\langle s_0, [0,0], 0\rangle,$$
$$\langle s_1, [0,1], 1\rangle,$$
$$\langle s_2, [3,0], 4\rangle)$$

The above sequence is *not* a run since it is finite.

seq_2

$$(\langle s_0, [0,0], 0\rangle,$$
$$\langle s_0, [t_i, t_i], t_i\rangle), \text{ where } t_i = \sum_{k=0}^{i} \frac{1}{2^k}, \text{ for all } i \geq 0.$$

In the above sequence, for all i, $t_i < 2$.

The above sequence is infinite but it is *not* a run because it does not satisfy the *progress* requirement of a run: for all $t \in R$, there exists i where $t_i \geq t$.

3.7. TCTL (Timed CTL) Syntax.

S1 $p \in AP$ is a TCTL formula

S2 If ϕ_1 and ϕ_2 are TCTL formulas, then so are $\phi_1 \wedge \phi_2$ and $\neg\phi_1$

S3 If ϕ_1 and ϕ_2 are TCTL formulas, then so are $A\phi_1 U_{\sim c}\phi_2$ and $E\phi_1 U_{\sim c}\phi_2$

Where $\sim \in \{<, \leq, =, \geq, >\}$ and $c \in N$

The class of formula generated by S1-S3 is the language of TCTL.

3.8. TCTL Semantics.

We assume that ρ is a $\langle s, \nu\rangle$-path of a timed transition system M based on a timed graph G, and $s = \langle s_0, \nu\rangle$ is a state in $S \times \Gamma(G)$.

S1 $M, s \models p$, iff $p \in \mu(s_0)$ for a $p \in AP$

S2 $M, s \models \phi_1 \wedge \phi_2$ iff $M, s \models \phi_1$ and $M, s \models \phi_2$
$M, s \models \neg\phi_1$ iff it is not the case that $M, s \models \phi_1$, for TCTL formulas ϕ_1 and ϕ_2

S3 $M, s \models E\phi_1 U_{\sim c}\phi_2$ iff for some path ρ, for some $t \sim c$, $M, \rho(t) \models \phi_2$, and for $0 \leq t' < t$, $M, \rho(t') \models \phi_1$

$M, s \models A\phi_1 U_{\sim c}\phi_2$ iff for all paths ρ, for some $t \sim c$, $M, \rho(t) \models \phi_2$, and for $0 \leq t' < t$, $M, \rho(t') \models \phi_1$

3.9. **Equivalence of Clock Assignments.**

For all $x \in C$, let c_x be the largest constant with which x is compared

Two clock assignments are equivalent ($\nu \cong \nu'$) iff:

- For each $x \in C$, $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$, or both $\nu(x)$ and $\nu'(x)$ are greater than c_x

- For each pair $x, y \in C$,
  s.t. $\nu(x) \leq c_x$ and $\nu(y) \leq c_y$,

  1. $fract(\nu(x)) \leq fract(\nu(y))$ iff
     $fract(\nu'(x)) \leq fract(\nu'(y))$

  2. $fract(\nu(x)) = 0$ iff $fract(\nu'(x)) = 0$

Our goal is to show that for equivalent clock assignment ν and ν', a TCTL formula ϕ, and $s \in S$, $M, \langle s, \nu \rangle \models \phi$ iff $M, \langle s, \nu' \rangle \models \phi$.

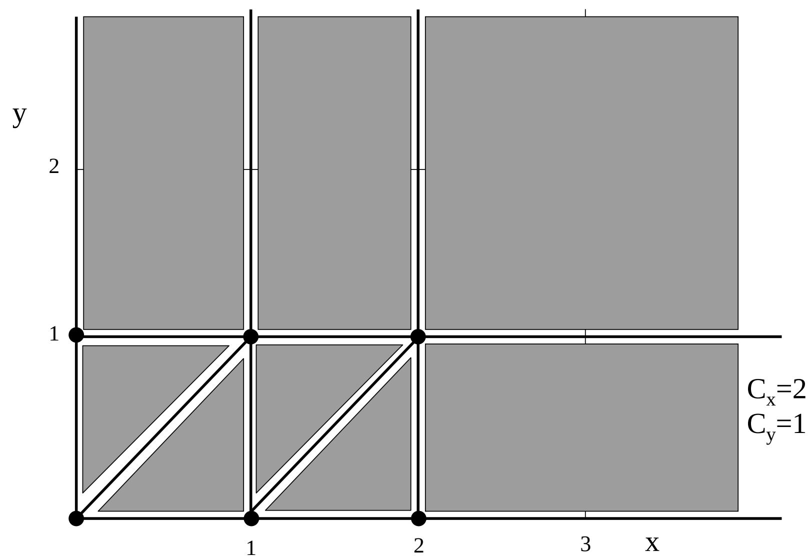

FIGURE 5. Equivalence Regions of Clocks $\{x, y\}$

3.10. **Successor Region.**

Let α be an equivalence class of the clock assignments $(\Gamma(G))$.

We denote that β is an equivalence class that is the successor of α, $\beta = Succ(\alpha)$, iff:

For a positive $t \in R$, and for $\nu \in \alpha$, $(\nu + t) \in \beta$, and for all $t' < t$, $(\nu + t') \in \alpha \cup \beta$.

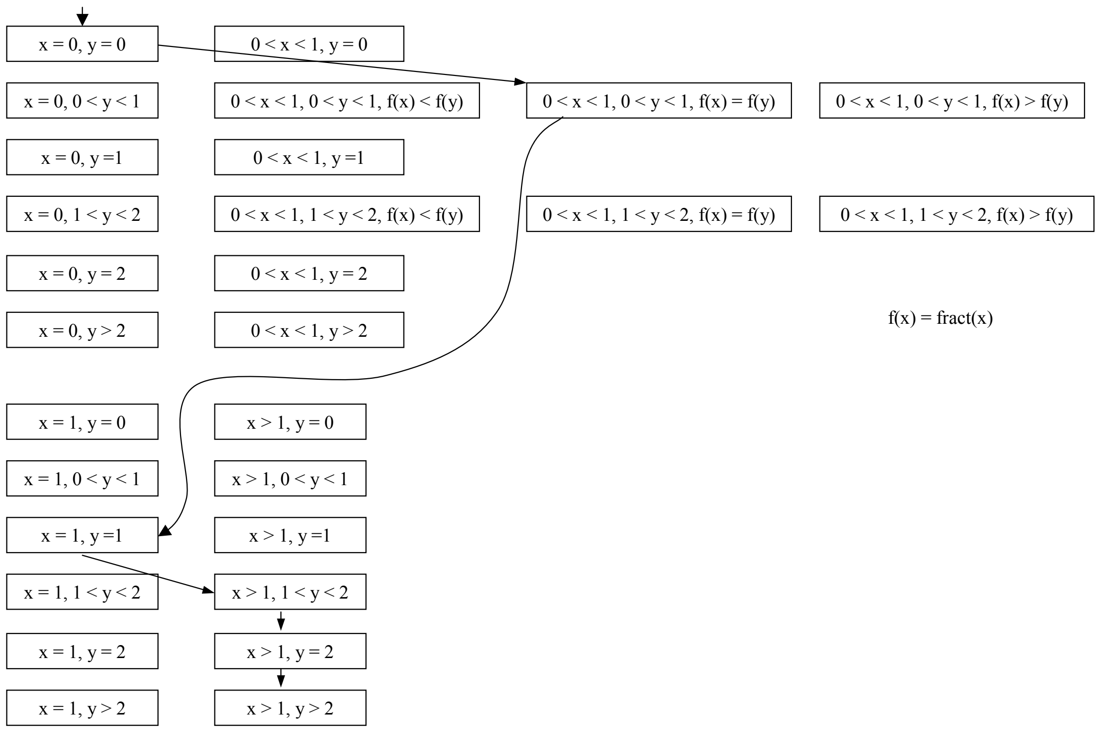

FIGURE 6. Example-1: Successor Regions $(c_x = 1, c_y = 2)$

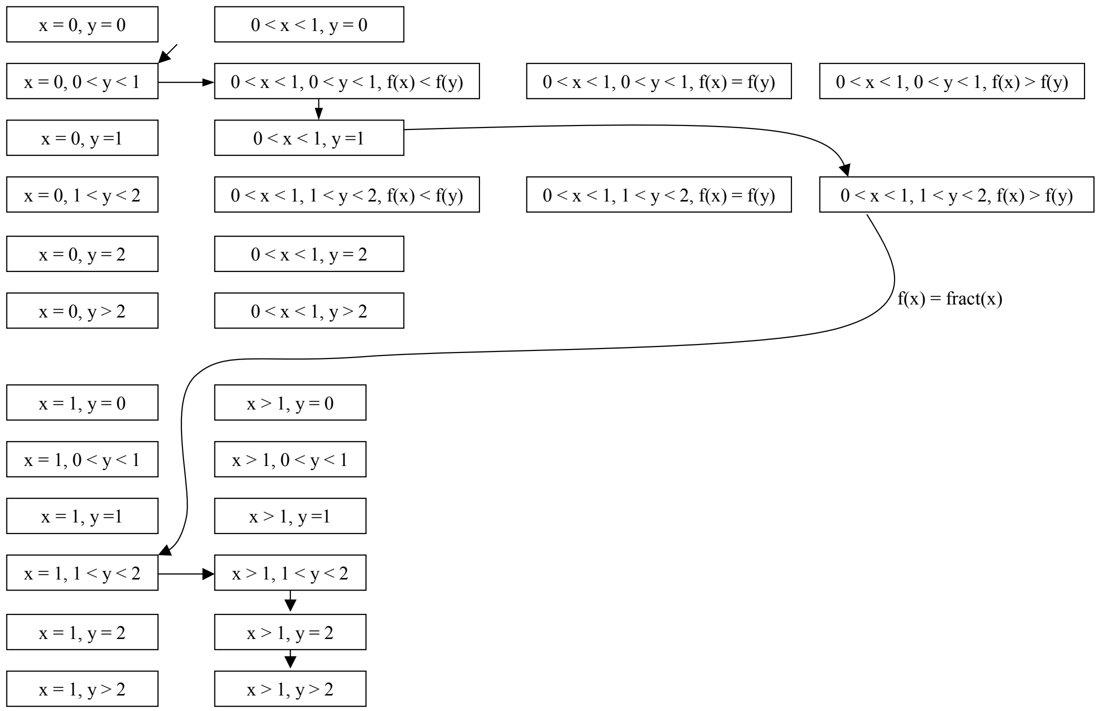

FIGURE 7. Example-2: Successor Regions ($c_x = 1, c_y = 2$)

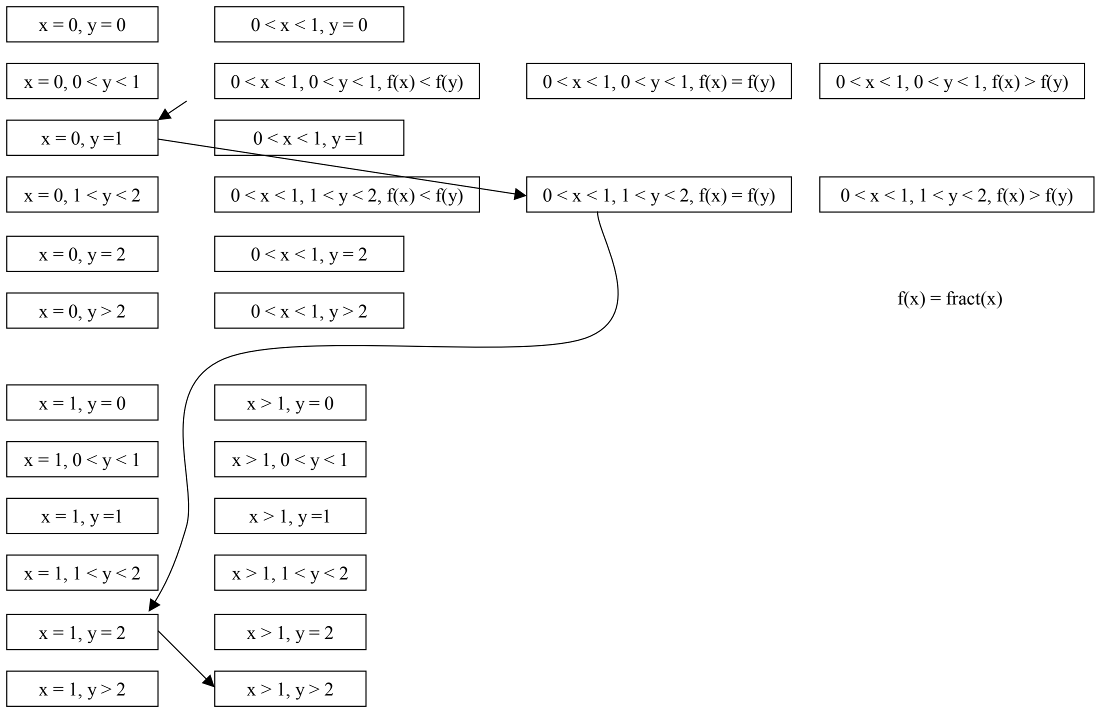

FIGURE 8. Example-3: Successor Regions $(c_x = 1, c_y = 2)$

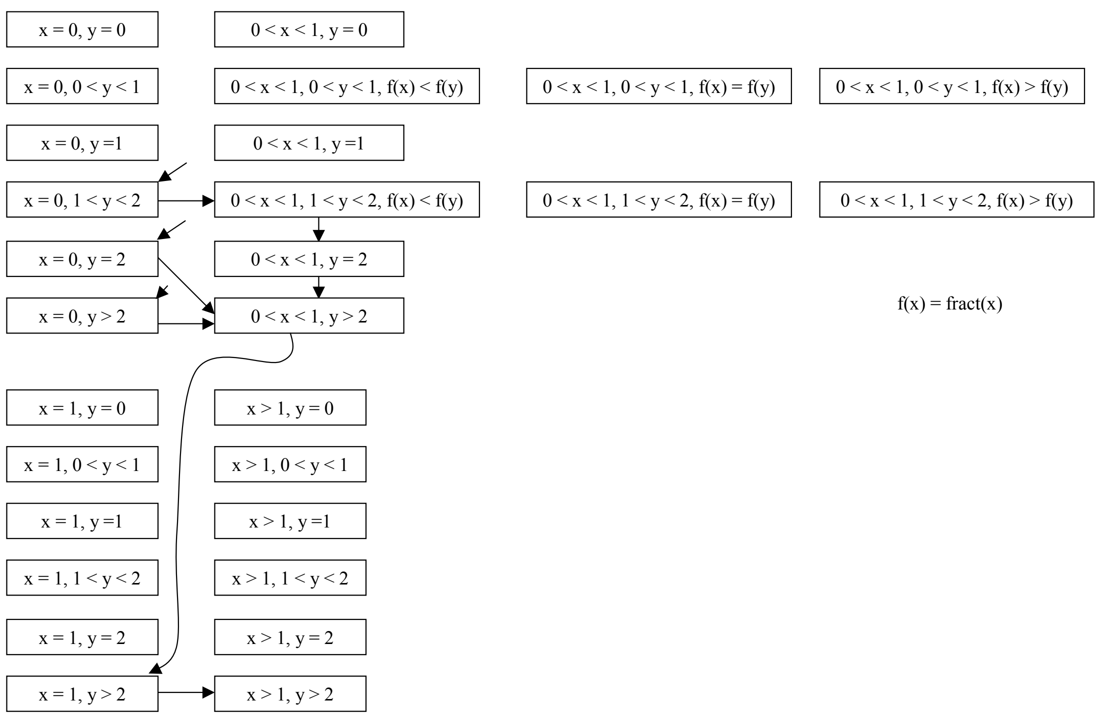

FIGURE 9. Example-4: Successor Regions ($c_x = 1, c_y = 2$)

3.11. Clock Regions vs. Augmented Clock Regions.

To the clock set C, add a clock x, not in C, that is not reset by any edge in the timed graph G. The clock regions resulting from the addition of x are called the *augmented clock regions*.

We denote by c_x the largest integer constant appearing in the TCTL formula.

The `augmented clock regions` refine a clock region due to the addition of the extra clock x.

Example clock region ...
$\{0 < y < 1\}$

... and its *augmented* clock regions (assume $c_x = 1$):
$\{0 < y < 1, x = 0\}$, $\{0 < y < 1, 0 < x < 1\}$,
$\{0 < y < 1, x = 1\}$, $\{0 < y < 1, x > 1\}$

We write C^* to represent the clock set with the added clock x.

We denote by $[\nu]^*$ the equivalence class with respect to the equivalence relation for clock assignments with clocks in C^*.

3.12. **Region Graph.**

The region graph consists of vertices V that is the product of the set of augmented regions with the nodes S of timed graph G.

The edges of the region graph are defined as follows;

`Edges representing the` *passage of time*: Each vertex $\langle s, \alpha \rangle$, where α is not an end class, has an edge to $\langle s, succ(\alpha) \rangle$

`Edges representing` *transitions in* G: Each vertex $\langle s, \alpha \rangle$ for each edge $e = \langle s, s' \rangle$, has an edge to $\langle s', [[\pi(e) \mapsto 0]\nu] \rangle$, provided that

i) α is not a boundary class*, and

ii) Either $\nu \in \alpha$ or $\nu \in succ(\alpha)$, and

iii) ν satisfies the enabling condition $\tau(e)$.

* A boundary class α is such that for a positive real t and all ν in α, $\nu + t$ is not equivalent to ν.

Examples:

$\{x = 0, 1 < y < 2\}$,
$\{x = 1, y = 2\}$

where $c_x = 1$, and $c_y = 2$.

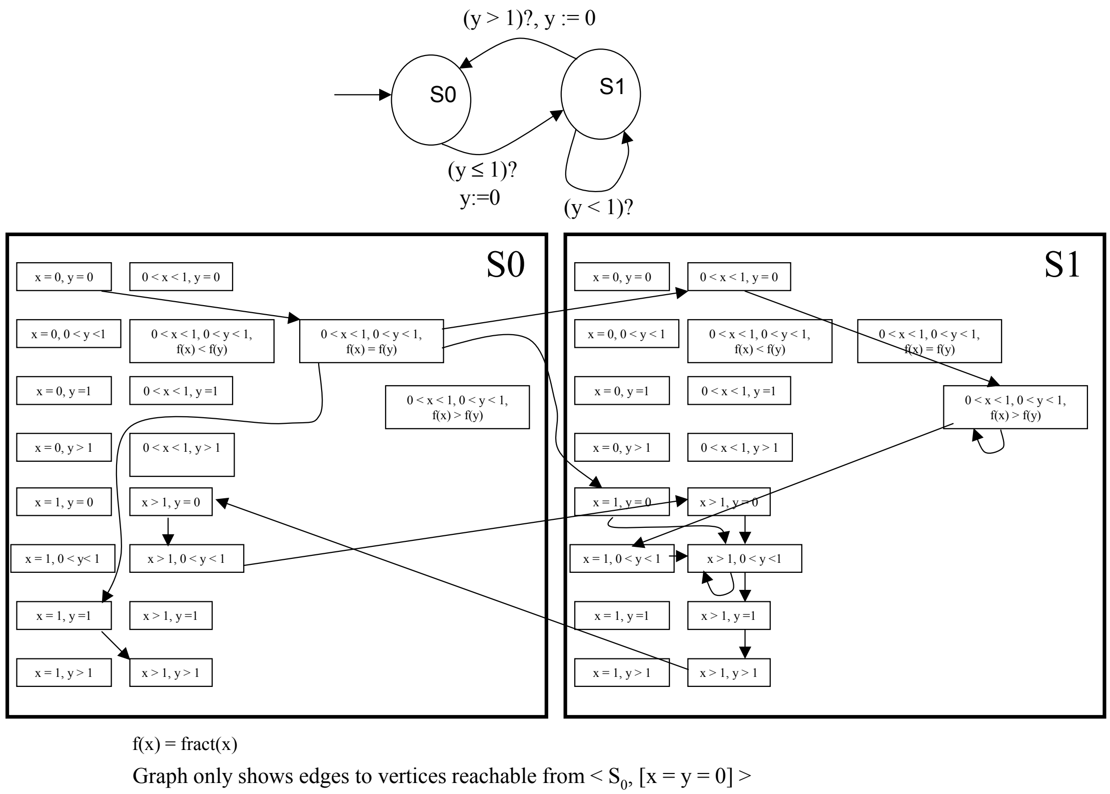

FIGURE 10. Timed Graph-1 and its Region Graph ($c_x = 1, c_y = 1$)

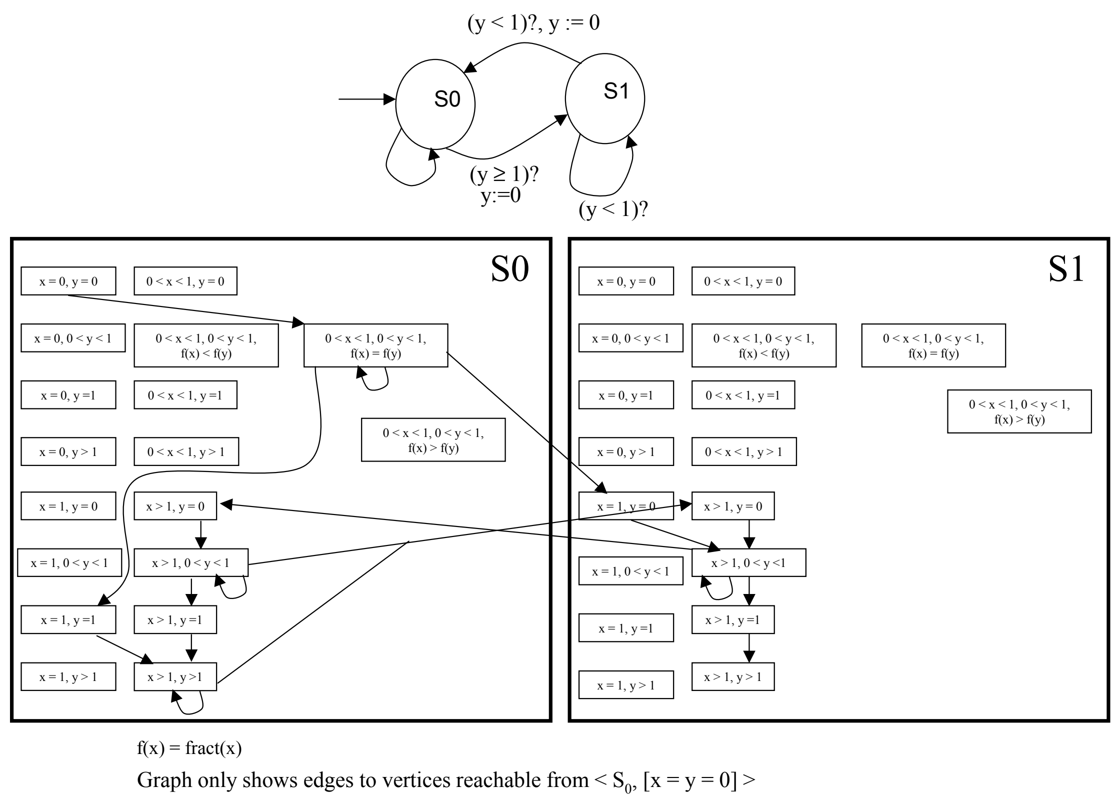

Figure 11. Timed Graph-2 and its Region Graph ($c_x = 1, c_y = 1$)

3.13. **Fair Paths in the Region Graph.**

- A path through the region graph is an infinite sequence of vertices in the region graph $\langle v_1, v_2, v_3, \ldots\rangle$, such that v_i has an edge to v_{i+1}.

- A path is fair if every clock in C^* is either reset infinitely often or is eventually always increasing.

- Hence, for all fair paths β through the region graph, for each clock $y \in C^*$, infinitely many vertices along the path β satisfy either $y = 0$, or $y > c_y$.

- In labelling the region graph, for each vertex v, for each clock $y \in C^*$, label vertex v with

  $p_{y=0}$ if $y = 0$ in v

  $p_{y>c_y}$ if $y > c_y$ in v

- Using Fair CTL, with clock set $C^* = \{x, y, z\}$, the fairness condition would be

  $$\overset{\infty}{\mathrm{F}}(p_{x=0} \vee p_{x>c_x}) \wedge \overset{\infty}{\mathrm{F}}(p_{y=0} \vee p_{y>c_y}) \wedge \overset{\infty}{\mathrm{F}}(p_{z=0} \vee p_{z>c_z}),$$

  Where $\overset{\infty}{\mathrm{F}}\, x$ denotes that the proposition x is true infinitely often along a path.

3.14. A Graph Labelling Algorithm.

For vertices in the region graph, every subscript $\sim c$ appearing in TCTL formula ϕ, label the vertex with $p_{\sim c}$ iff at vertex $\langle s, [\nu]^* \rangle$, $\nu \models x \sim c$.

Also label vertices with P_b if a vertex represents a boundary class.

For a formula of the form $EpU_{\sim c}q$, where p and q are propositions, label $v = \langle s, [\nu]^* \rangle^\dagger$ with ϕ iff:

For some fair path starting at $\langle s, [[x \mapsto 0]\nu]^* \rangle$,

Has a prefix $(v_1, v_2, v_3, \ldots)$ such that

- For each $i \leq n$, v_i is labelled with p, and
- v_n is labelled with q and
- v_n is labelled with $p \sim c$, and
- v_n is either labelled with p_b or p.

† When labelling a vertex $\langle s, [\nu]^* \rangle$ with ϕ, where $[\nu]^*$ is a refinement of a clock region α, we also label $\langle s, [\nu']^* \rangle$ with ϕ, where $[\nu']^*$ ($\neq [\nu]^*$) is a refinement of the same clock region α.

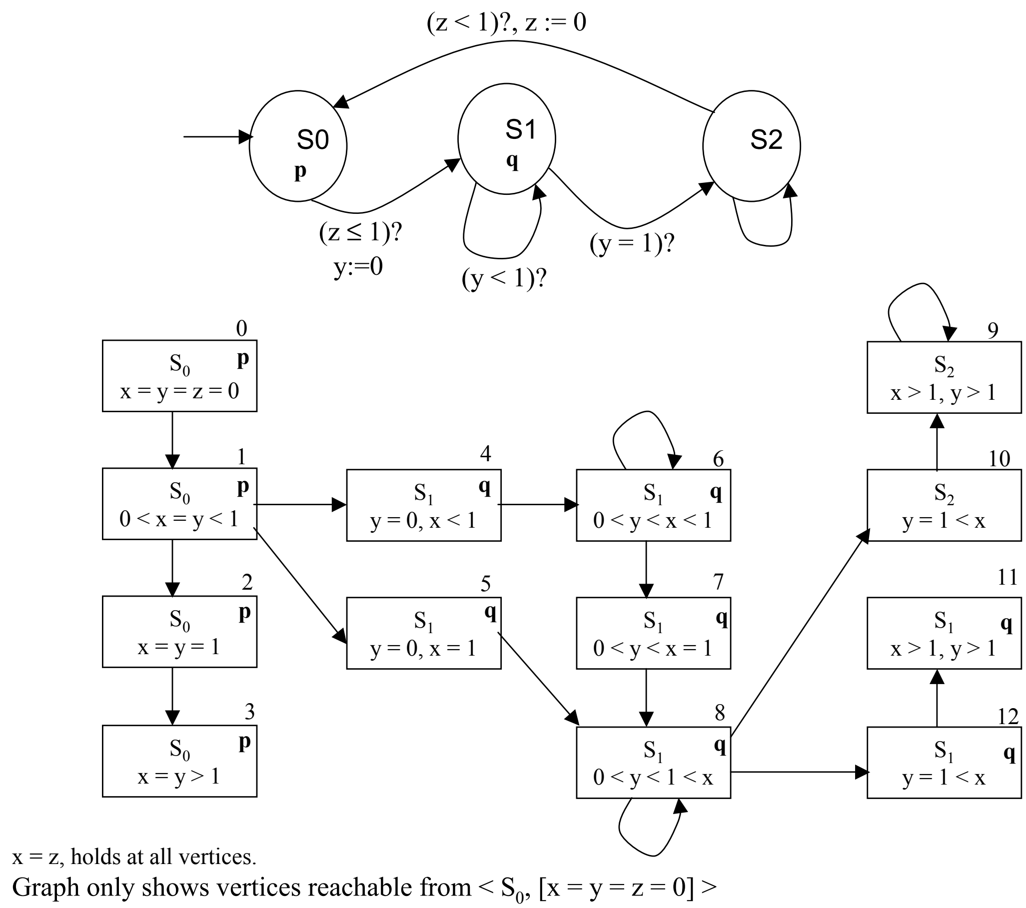

Figure 12. Example TCTL Model-Checking

3.15. A Procedure Using Fair CTL to Model-Check a Region Graph.

- Remove vertices, and associated edges, from the region graph that do not have an outgoing edge (repeat this step until all such vertices are removed).

- For vertices in the region graph, every subscript $\sim c$ appearing in TCTL formula ϕ, label the vertex with $p_{\sim c}$ iff at vertex $\langle s, [\nu]^* \rangle$, $\nu \models x \sim c$.

- Also label vertices with P_b if a vertex represents a boundary class.

- For a TCTL formula ϕ of the form

  $E\phi_1 U_{\sim c} \phi_2,$

  we use the Fair CTL formula ϕ' of the form

  $E\phi_1 U p_c \wedge \phi_2 \wedge (p_b \vee \phi_1)$

- We assume that all TCTL subformulas ϕ_1 and ϕ_2 of TCTL formula ϕ have already been checked using this procedure. (i.e., the graph is already labelled with ϕ_1 and ϕ_2)

- For each vertex v, for each clock $y \in C^*$, label vertex v with

  $p_{y=0}$ if $y = 0$ in v

  $p_{y>c_y}$ if $y > c_y$ in v

- Using Fair CTL, with clock set C^*, the fairness condition is

$$\bigwedge_{y \in C^*} \overset{\infty}{\mathrm{F}}(p_{y=0} \vee p_{y>c_y})$$

- We assume that the Fair CTL Model-Checker returns the set of vertices $S_{\phi'}$ that satisfy the given formula ϕ', but does not label the graph with ϕ'.

- Remove those vertices from $S_{\phi'}$ where $x \neq 0$.

- For the vertices that remain in $S_{\phi'}$, label each vertex with ϕ.

- When labelling a vertex $\langle s, [\nu]^* \rangle$ with ϕ, where $[\nu]^*$ is a refinement of a clock region α, we also label $\langle s, [\nu']^* \rangle$ with ϕ, where $[\nu']^*$ ($\neq [\nu]^*$) is a refinement of the same clock region α.

Department of Electrical & Computer Engineering University of Texas at Austin
E-mail address: `shant@mail.utexas.edu`