Online appendix for the paper

Representing Hybrid Automata by Action Language Modulo Theories

published in Theory and Practice of Logic Programming

Joohyung Lee, Nikhil Loney
School of Computing, Informatics and Decision Systems Engineering
Arizona State University, Tempe, AZ, USA

Yunsong Meng
Houzz, Inc.
Palo Alto, CA, USA

Appendix A Review: $\mathcal{C}$+

A.1 Syntax of $\mathcal{C}$+

$\mathcal{C}$+ was originally defined as a propositional language (Giunchiglia et al. 2004). In this section we review its reformulation in terms of ASPMT (Lee and Meng 2013).

We consider a many-sorted first-order signature σ that is partitioned into three sub-signatures: the set σ^{fl} of object constants called *fluent constants*, the set σ^{act} of object constants called *action constants*, and the background signature σ^{bg}. The signature σ^{fl} is further partitioned into the set σ^{sim} of *simple* fluent constants and the set σ^{sd} of *statically determined* fluent constants.

A *fluent formula* is a formula of signature $\sigma^{fl} \cup \sigma^{bg}$. An *action formula* is a formula of $\sigma^{act} \cup \sigma^{bg}$ that contains at least one action constant and no fluent constants.

A *static law* is an expression of the form

$$\textbf{caused } F \textbf{ if } G \tag{A1}$$

where F and G are fluent formulas.

An *action dynamic law* is an expression of the form (A1) in which F is an action formula and G is a formula.

A *fluent dynamic law* is an expression of the form

$$\textbf{caused } F \textbf{ if } G \textbf{ after } H \tag{A2}$$

where F and G are fluent formulas and H is a formula, provided that F does not contain statically determined constants.

A *causal law* is a static law, an action dynamic law, or a fluent dynamic law. An *action description* is a finite set of causal laws.

The formula F in causal laws (A1) and (A2) is called the *head*.

We call an action description *definite* if the head F of every causal law (A1) and (A2) is an atomic formula that is $(\sigma^{fl} \cup \sigma^{act})$-plain. [8]

[8] For any function constant f, we say that a first-order formula is f*-plain* if each atomic formula in it

- does not contain f, or
- is of the form $f(\mathbf{t}) = t_1$ where $\mathbf{t}$ is a list of terms not containing f, and t_1 is a term not containing f.

For any list $\mathbf{c}$ of predicate and function constants, we say that F is $\mathbf{c}$-plain if F is f-plain for each function constant f in $\mathbf{c}$.

A.2 Semantics of $\mathcal{C}$+

For a signature σ and a nonnegative integer i, expression $i:\sigma$ is the signature consisting of the pairs $i:c$ such that $c \in \sigma$, and the value sort of $i:c$ is the same as the value sort of c. Similarly, if s is an interpretation of σ, expression $i:s$ is an interpretation of $i:\sigma$ such that $c^s = (i:c)^{i:s}$.

For any action description D of signature $\sigma^{fl} \cup \sigma^{act} \cup \sigma^{bg}$ and any nonnegative integer m, the ASPMT program D_m is defined as follows. The signature of D_m is $0\!:\!\sigma^{fl} \cup \dots \cup m\!:\!\sigma^{fl} \cup 0\!:\!\sigma^{act} \cup \dots \cup (m{-}1)\!:\!\sigma^{act} \cup \sigma^{bg}$. By $i:F$ we denote the result of inserting $i:$ in front of every occurrence of every fluent and action constant in a formula F.

ASPMT program D_m is the conjunction of

$$i\!:\!G \rightarrow i\!:\!F$$

for every static law (A1) in D and every $i \in \{0, \dots, m\}$, and for every action dynamic law (A1) in D and every $i \in \{0, \dots, m{-}1\}$;

$$(i{+}1)\!:\!G \wedge i\!:\!H \rightarrow (i{+}1)\!:\!F$$

for every fluent dynamic law (A2) in D and every $i \in \{0, \dots, m-1\}$.

The transition system represented by an action description D consists of states (vertices) and transitions (edges). A *state* is an interpretation s of σ^{fl} such that $0\!:\!s \models_{bg} \text{SM}[D_0;\ 0{:}\sigma^{sd}]$. A *transition* is a triple $\langle s, e, s' \rangle$, where s and s' are interpretations of σ^{fl} and e is an interpretation of σ^{act}, such that

$$(0\!:\!s) \cup (0\!:\!e) \cup (1\!:\!s') \models_{bg} \text{SM}[D_1;\ (0\!:\!\sigma^{sd}) \cup (0\!:\!\sigma^{act}) \cup (1\!:\!\sigma^{fl})]\ .$$

The definition of the transition system above implicitly relies on the following property of transitions:

Theorem 2
(Lee and Meng 2013, Theorem 3) For every transition $\langle s, e, s' \rangle$, s and s' are states.

The following theorem states the correspondence between the stable models of D_m and the paths in the transition system represented by D:

Theorem 3
(Lee and Meng 2013, Theorem 4)

$$\begin{aligned}&(0\!:\!s_0) \cup (0\!:\!e_0) \cup (1\!:\!s_1) \cup (1\!:\!e_1) \cup \dots \cup (m\!:\!s_m)\\ &\quad \models_{bg} \text{SM}[D_m;\ (0\!:\!\sigma^{sd}) \cup (0\!:\!\sigma^{act}) \cup (1\!:\!\sigma^{fl}) \cup (1\!:\!\sigma^{act}) \cup \dots \cup (m{-}1\!:\!\sigma^{act}) \cup (m\!:\!\sigma^{fl})]\end{aligned}$$

iff each triple $\langle s_i, e_i, s_{i+1} \rangle$ $(0 \leq i < m)$ is a transition.

It is known that when D is definite, ASPMT program D_m that is obtained from action description D is always tight. Functional completion (Bartholomew and Lee 2013) on ASPMT can be applied to turn D_m into an SMT instance.

A.3 Some Useful Abbreviations of $\mathcal{C}$+ Causal Laws

This section explains the abbreviations of $\mathcal{C}$+ causal laws used in the paper.

1. A static law of the form

$$\textbf{caused}\ \bot\ \textbf{if}\ \neg F$$

can be written as

$$\textbf{constraint } F.$$

2. A fluent dynamic law of the form

$$\textbf{caused } \bot \textbf{ if } \neg F \textbf{ after } G$$

can be written as

$$\textbf{constraint } F \textbf{ after } G.$$

3. A fluent dynamic law of the form

$$\textbf{caused } \bot \textbf{ after } F \wedge G$$

where F is an action formula can be written as

$$\textbf{nonexecutable } F \textbf{ if } G. \tag{A3}$$

4. An expression of the form

$$F \textbf{ causes } G \textbf{ if } H \tag{A4}$$

where F is an action formula stands for the fluent dynamic law

$$\textbf{caused } G \textbf{ after } F \wedge H$$

if G is a fluent formula,[9] and for the action dynamic law

$$\textbf{caused } G \textbf{ if } F \wedge H$$

if G is an action formula.

5. An expression of the form

$$\textbf{default } F \textbf{ if } G \tag{A5}$$

stands for the causal law

$$\textbf{caused } \{F\}^{\text{ch}} \textbf{ if } G.$$

6. An expression of the form

$$\textbf{default } F \textbf{ if } G \textbf{ after } H$$

stands for the fluent dynamic law [10]

$$\textbf{caused } \{F\}^{\text{ch}} \textbf{ if } G \textbf{ after } H.$$

7. An expression of the form

$$\textbf{exogenous } c \textbf{ if } G \tag{A6}$$

where c is a constant stands for the set of causal laws

$$\textbf{default } c\!=\!v \textbf{ if } G$$

[9] It is clear that the expression in the previous line is a fluent dynamic law only when G does not contain statically determined fluent constants. Similar remarks can be made in connection with many of the abbreviations introduced below.

[10] $\{F\}^{\text{ch}}$ stands for choice formula $F \vee \neg F$.

for all $v \in \mathit{Dom}(c)$.

8. An expression of the form

$$\textbf{inertial}\ c\ \textbf{if}\ G \tag{A7}$$

where c is a fluent constant stands for the set of fluent dynamic laws

$$\textbf{default}\ c\!=\!v\ \textbf{after}\ c\!=\!v \wedge G$$

for all $v \in$ *Dom(c)*.

9. In the abbreviations of causal laws above, ”**if** G” and ”**if** H” can be omitted if G and H are $\top$.

Appendix B Proofs

B.1 Proof of Theorem 1

We assume the case for linear hybrid automata with convex invariants. The proof of the general case of non-linear hybrid automata with non-convex invariants are mostly similar except for the difference in Flow and Inv conditions.

Theorem 1
There is a 1:1 correspondence between the paths of the transition system of a Hybrid automata H and the paths of the transition system of the $\mathcal{C}+$ action description D_H.

The proof is immediate from Lemma 1 and Lemma 2, which are proven below.

B.1.1 Proof of Lemma 1

Lemma 3
Let H be a linear hybrid automaton with convex invariants, and let

$$(v, r) \xrightarrow{\sigma} (v, r')$$

be a transition in T_H such that $\sigma \in \mathcal{R}_{>0}$. Function $f(t)\!=\!r+t\!\times\!(r'\!-\!r)/\sigma$ is a linear differentiable function from $[0, \sigma]$ to $\mathcal{R}^n$, with the first derivative $\dot{f} : [0, \sigma] \rightarrow \mathcal{R}^n$ such that (i) $f(0) = r$ and $f(\sigma)\!=\!r'$ and (ii) for all reals $\epsilon \in (0, \sigma)$, both $\mathsf{Inv}_v(f(\epsilon))$ and $\mathsf{Flow}_v(\dot{f}(\epsilon))$ are true.

Proof. We check that f satisfies the above conditions:

- It is clear that $f(t)$ is differentiable over $t \in [0, \sigma]$, $f(0)\!=\!r$ and $f(\sigma)\!=\!r'$.
- Since (v, r) and (v, r') are states of T_H, it follows that $\mathsf{Inv}_v(f(0))$ and $\mathsf{Inv}_v(f(\sigma))$ are true. Since the values of X that makes $\mathsf{Inv}_v(X)$ form a convex region in $\mathcal{R}^n$ and $f(t)$ is a linear function, it follows that for $\epsilon \in (0, \sigma)$, $\mathsf{Inv}_v(f(\epsilon))$ is true.
- Since $(v, r) \xrightarrow{\sigma} (v, r')$ is a transition in T_H, it follows that there is a function g such that (i) g is differentiable in $[0, \sigma]$, (ii) for any $\epsilon \in (0, \sigma)$, $\mathsf{Flow}_v(\dot{g}(\epsilon))$ is true, (iii) $g(0) = r$ and $g(\sigma) = r'$. Since g is continuous on $[0, \sigma]$ (differentiability implies continuity) and differentiable on $(0, \sigma)$, by the mean value theorem[11], there is a point $c \in (0, \sigma)$ such that $\dot{g}(c) = (r'\!-\!r)/\sigma$. Consequently, $\mathsf{Flow}_v((r'\!-\!r)/\sigma)$ is true. As a result, we get $\mathsf{Flow}_v(\dot{f}(\epsilon))$ is true for all $\epsilon \in (0, \sigma)$.

□

In the following two lemmas, s_i, a_i, s_{i+1} are defined as in Lemma 1.

[11] http://en.wikipedia.org/wiki/Mean_value_theorem

Lemma 4
For each $i \geq 0$, s_i is a state in the transition system of D_H.

Proof. Since D_H does not contain statically determined fluent constants and every simple fluent constant is declared exogenous, it is sufficient to prove

$$0\!:\!s_i \models_{bg} \text{SM}[(D_H)_0;\ \emptyset],$$

while $\text{SM}[(D_H)_0;\ \emptyset]$ is equivalent to the conjunction of

$$0\!:\!\mathit{Mode} = v \rightarrow 0\!:\!\mathsf{Inv}_v(X) \tag{B1}$$

for each $v \in V$. Since p is a path, for each $i \geq 0$, (v_i, r_i) is a state in T_H. By the definition of a hybrid transition system, $\mathsf{Inv}_{v_i}(r_i)$ is true. Since $s_i \models_{bg} (\mathit{Mode}, X) = (v_i, r_i)$, we have $0\!:\!s_i \models_{bg}$ (B1). $\square$

Lemma 5
For each $i \geq 0$, $\langle s_i, a_i, s_{i+1}\rangle$ is a transition in the transition system of D_H.

Proof. By definition, we are to show that

$$0\!:\!s_i \cup 0\!:\!a_i \cup 1\!:\!s_{i+1} \models_{bg} \text{SM}[(D_H)_1; 0\!:\!\sigma^{act} \cup 1\!:\!\sigma^{fl}]. \tag{B2}$$

We check that $(D_H)_1$ is tight, so that (B2) is equivalent to

$$0\!:\!s_i \cup 0\!:\!a_i \cup 1\!:\!s_{i+1} \models_{bg} \text{Comp}[(D_H)_1; 0\!:\!\sigma^{act} \cup 1\!:\!\sigma^{fl}],$$

where the completion $\text{Comp}[(D_H)_1; 0 : \sigma^{act} \cup 1 : \sigma^{fl}]$ is equivalent to the conjunction of the following formulas:

- Formula *FLOW*, which is the conjunction of

  $$\mathsf{Flow}_v((1\!:\!X - 0\!:\!X)/t) \leftarrow 0\!:\!\mathit{Mode} = v \wedge 0\!:\!\mathit{Dur} = t \wedge 0\!:\!\mathit{Wait} = \text{TRUE} \wedge t > 0 \tag{B3}$$

  and

  $$1\!:\!X = 0\!:\!X \ \leftarrow 0\!:\!\mathit{Mode} = v \ \wedge \ 0\!:\!\mathit{Dur} = 0 \ \wedge \ 0\!:\!\mathit{Wait} = \text{TRUE} \tag{B4}$$

  for each $v \in V$.
- Formula *INV*, which is the conjunction of

  $$k : \mathsf{Inv}_v(X) \ \leftarrow \ k : \mathit{Mode} = v \tag{B5}$$

  for each $k \in \{0, 1\}$ and each $v \in V$.
- Formula *WAIT*, which is the conjunction of

  $$0\!:\!\mathit{Wait} = \text{FALSE} \ \leftrightarrow \ \bigvee_{e \in E} 0\!:\!\mathsf{hevent}(e) = \text{TRUE}.$$

- Formula *GUARD*, which is the conjunction of

  $$\bot \ \leftarrow \ 0\!:\!\mathsf{hevent}(e) = \text{TRUE} \ \wedge \ 0\!:\!\neg\mathsf{Guard}_e(X) \tag{B6}$$

  for each edge $e \in E$.
- Formula *RESET*, which is the conjunction of

  $$\mathsf{Reset}_e(0\!:\!X, 1\!:\!X) \ \leftarrow \ 0\!:\!\mathsf{hevent}(e) = \text{TRUE}$$

  for each edge $e = (v_1, v_2) \in E$.

- Formula *MODE*, which is the conjunction of

  $$\bot \leftarrow 0\!:\!\mathsf{hevent}(e)\!=\!\text{TRUE} \;\wedge\; \neg(0\!:\!\mathit{Mode} = v_1)$$

  for each $e = (v_1, v_2) \in E$;

  $$1 : \mathit{Mode} = v \;\leftrightarrow\; \bigvee_{\{v' \mid (v',v) \in E\}} 0\!:\!\mathsf{hevent}(v', v)\!=\!\text{TRUE} \vee\; 0\!:\!\mathit{Mode} = v$$

  for each $v \in V$.
- Formula *DURATION*, which is the conjunction of

  $$0\!:\!\mathit{Dur} = 0 \;\leftarrow\; \bigvee_{e \in E} 0\!:\!\mathsf{hevent}(e)$$

  for each edge $e \in E$.

We will show that $0\!:\!s_i \cup 0\!:\!a_i \cup 1\!:\!s_{i+1}$ satisfies each of the formulas above. First, we check *INV*.

- *INV*: From the fact that (v_i, r_i) and (v_{i+1}, r_{i+1}) are states in T_H, by the definition of a hybrid transition system, $\mathsf{Inv}_{v_i}(r_i)$ and $\mathsf{Inv}_{v_{i+1}}(r_{i+1})$ are true. Note that $s_i \models_{bg} (\mathit{Mode}, X) = (v_i, r_i)$ and $s_{i+1} \models_{bg} (\mathit{Mode}, X) = (v_{i+1}, r_{i+1})$. As a result,

  $$\begin{array}{l} 0\!:\!s_i \models_{bg} (0\!:\!\mathit{Mode} = v \;\rightarrow\; 0\!:\!\mathsf{Inv}_{v_i}(X)) \\ 1\!:\!s_{i+1} \models_{bg} (1\!:\!\mathit{Mode} = v \;\rightarrow\; 1\!:\!\mathsf{Inv}_{v_{i+1}}(X)). \end{array}$$

  Hence $0\!:\!s_i \cup 0\!:\!a_i \cup 1\!:\!s_{i+1} \models_{bg} \mathit{INV}$.

Next, we check the remaining formulas. From the definition of T_H, there are two cases for the value of σ_i.

Case 1: $\sigma_i = \mathsf{hevent}(e)$ where $e = (v_i, v_{i+1})$. It follows from the construction of p' that $(\mathit{Dur})^{a_i} = 0$, $(\mathsf{hevent}(e))^{a_i} = \text{TRUE}$, $(\mathsf{hevent}(e'))^{a_i} = \text{FALSE}$ for all $e' \neq e$ and $(\mathit{Wait})^{a_i} = \text{FALSE}$.

From the fact that

$$(v_i, r_i) \xrightarrow{\sigma_i} (v_{i+1}, r_{i+1})$$

is a transition in T_H and that $\sigma_i = \mathsf{hevent}(e)$, it follows from the definition of a hybrid transition system that $\mathsf{Guard}_e(r_i)$ and $\mathsf{Reset}_e(r_i, r_{i+1})$ are true.

- *FLOW*: Since $0 : a_i \models 0 : \mathit{Wait} = \text{FALSE}$, trivially, $0\!:\!s_i \;\cup\; 0\!:\!a_i \;\cup\; 1\!:\!s_{i+1} \models_{bg} \mathit{FLOW}$.
- *WAIT*: Since $(\mathsf{hevent}(e))^{a_i} = \text{TRUE}$, and $(\mathit{Wait})^{a_i} = \text{FALSE}$, it follows that $0 : s_i \;\cup\; 0 : a_i \;\cup\; 1\!:\!s_{i+1} \models_{bg} \mathit{WAIT}$.
- *GUARD*: From $s_i \models_{bg} X = r_i$, it follows that $0\!:\!s_i \models_{bg} 0\!:\!\mathsf{Guard}_e(X)$. Since $(\mathsf{hevent}(e))^{a_i} = \text{TRUE}$, it follows that $0\!:\!s_i \;\cup\; 0\!:\!a_i \;\cup\; 1\!:\!s_{i+1} \models_{bg} \mathit{GUARD}$.
- *RESET*: From $s_i \models_{bg} (\mathit{Mode}, X) = (v_i, r_i)$ and $s_{i+1} \models_{bg} (\mathit{Mode}, X) = (v_{i+1}, r_{i+1})$, it follows that $0 : s_i \;\cup\; 1 : s_{i+1} \models_{bg} \mathsf{Reset}_e(0 : X, 1 : X)$. Since $(\mathsf{hevent}(e))^{a_i} = \text{TRUE}$, it follows that $0\!:\!s_i \;\cup\; 0\!:\!a_i \;\cup\; 1\!:\!s_{i+1} \models_{bg} \mathit{RESET}$.
- *MODE*: Note that $s_i \models_{bg} (\mathit{Mode}, X) = (v_i, r_i)$ and $s_{i+1} \models_{bg} (\mathit{Mode}, X) = (v_{i+1}, r_{i+1})$. It is immediate that $0 : s_i \models_{bg} 0 : \mathit{Mode} = v_i$ and $1 : s_{i+1} \models_{bg} 1 : \mathit{Mode} = v_{i+1}$. Since $(\mathsf{hevent}(e))^{a_i} = \text{TRUE}$, it follows that $0\!:\!s_i \;\cup\; 0\!:\!a_i \;\cup\; 1\!:\!s_{i+1} \models_{bg} \mathit{MODE}$.
- *DURATION*: Since $(\mathit{Dur})^{a_i} = 0$ and $(\mathsf{hevent}(e))^{a_i} = \text{TRUE}$, it follows that $0 : s_i \;\cup\; 0 : a_i \;\cup\; 1\!:\!s_{i+1} \models_{bg} \mathit{DURATION}$.

Case 2: $\sigma_i \in \mathcal{R}_{\geq 0}$. By the construction of p', $(\mathit{Dur})^{a_i} = \sigma_i$, $(\mathit{Wait})^{a_i} = \text{TRUE}$ and $(\mathsf{hevent}(e))^{a_i} = \text{FALSE}$ for every $e = (v, v') \in E$. It is easy to check that *WAIT*, *GUARD*, *RESET*, *MODE*, *DURATION* are trivially satisfied by $0\!:\!s_i \cup 0\!:\!a_i \cup 1\!:\!s_{i+1}$. So, it is sufficient to consider only *FLOW*.

From the fact that

$$(v_i, r_i) \xrightarrow{\sigma_i} (v_{i+1}, r_{i+1})$$

is a transition of T_H and that $\sigma_i \in \mathcal{R}_{\geq 0}$, it follows from the definition of a hybrid transition system that

(a) $v_i = v_{i+1}$, and

(b) there is a differentiable function $f : [0, \sigma_i] \rightarrow \mathcal{R}^n$, with the first derivative $\dot{f} : [0, \sigma_i] \rightarrow \mathcal{R}^n$ such that: (1) $f(0) = r_i$ and $f(\sigma_i) = r_{i+1}$ and (2) for all reals $\epsilon \in (0, \sigma_i)$, both $\mathsf{Inv}_{v_i}(f(\epsilon))$ and $\mathsf{Flow}_{v_i}(\dot{f}(\epsilon))$ are true.

- *FLOW*:

  — If $\sigma_i = 0$, then $(\mathit{Dur})^{a_i} = 0$. From (b), $r_i = r_{i+1} = f(0)$. As a result $X^{s_i} = X^{s_{i+1}}$ and it follows that $0\!:\!s_i \cup 0\!:\!a_i \cup 1\!:\!s_{i+1} \models_{bg} (B4)$.

  — If $\sigma_i > 0$, then $(\mathit{Dur})^{a_i} > 0$. By Lemma 3, $f(t) = r_i + t * (r_{i+1} - r_i)/\sigma_i$ is a differentiable function that satisfies all the conditions in (b). As a result, $\mathsf{Flow}_{v_i}((r_{i+1} - r_i)/\sigma_i)$ is true and thus $0 : s_i \cup 0 : a_i \cup 1 : s_{i+1} \models_{bg} \mathsf{Flow}_{v_i}((1\!:\!r - 0\!:\!r)/\mathit{Dur})$. It follows that $0\!:\!s_i \cup 0\!:\!e_i \cup 1\!:\!s_{i+1} \models_{bg} (B3)$.

□

Lemma 1
p' is a path in the transition system D_H.

Proof. By Lemma 4, each s_i is a state of D_H. By Lemma 5, each $\langle s_i, a_i, s_{i+1}\rangle$ is a transition of D_H. So p' is a path in the transition system of D_H. □

B.1.2 Proof of Lemma 2

In the following two lemmas, v_i, r_i are defined as in Lemma 2.

Lemma 6
For each $i \geq 0$, (v_i, r_i) is a state in T_H.

Proof. By definition, we are to show that $\mathsf{Inv}_{v_i}(r_i)$ is true. Since each s_i is a state in the transition system of D_H, by definition,

$$0\!:\!s_i \models_{bg} \text{SM}[(D_H)_0; \emptyset]. \tag{B7}$$

Note that $\text{SM}[(D_H)_0; \emptyset]$ is equivalent to the conjunction of the formula:

$$0\!:\!\mathsf{Inv}_v(X) \leftarrow 0\!:\!\mathit{Mode} = v \tag{B8}$$

for each $v \in V$. Since $(\mathit{Mode})^{s_i} = v_i$, it follows that $s_i \models_{bg} \mathsf{Inv}_{v_i}(X)$. Since $X^{s_i} = r_i$, it follows that $\mathsf{Inv}_{v_i}(r_i)$ is true. □

Lemma 7
For each $i \geq 0$, $(v_i, r_i) \xrightarrow{\sigma_i} (v_{i+1}, r_{i+1})$ is a transition in T_H.

Proof. From the fact that (s_i, a_i, s_{i+1}) is a transition of D_H, by definition we know that

$$0{:}s_i \cup 0{:}a_i \cup 1{:}s_{i+1} \models_{bg} \text{SM}[(D_H)_1;\ 0{:}\sigma^{act} \cup 1{:}\sigma^{fl}]. \tag{B9}$$

Since $(D_H)_1$ is tight, $\text{SM}[(D_H)_1; 0{:}\sigma^{act} \cup 1{:}\sigma^{fl}]$ is equivalent to $\text{Comp}[(D_H)_1; 0{:}\sigma^{act} \cup 1{:}\sigma^{fl}]$, which is equivalent to the conjunction of ***FLOW***, ***INV***, ***WAIT***, ***GUARD***, ***RESET***, ***MODE***, ***DURATION*** (See the proof of Lemma 5 for the definitions of these formulas).

Consider two cases:

Case 1: There exists an edge $e = (v, v')$ such that $(\mathsf{hevent}(e))^{a_i} = \text{TRUE}$. Since $\textit{Mode}^{s_i} = v_i$ and $\textit{Mode}^{s_{i+1}} = v_{i+1}$, it follows that (v, v') must be (v_i, v_{i+1}). Since $(\mathsf{hevent}(e))^{a_i} = \text{TRUE}$, it follows from the definition that σ_i is $\mathsf{hevent}(e)$.

- Since $0{:}s_i \cup 0{:}a_i \cup 1{:}s_{i+1} \models_{bg}$ ***GUARD*** and $X^{s_i} = r_i$, it is immediate that $\mathsf{Guard}_e(r_i)$ is true.
- Since $0{:}s_i \cup 0{:}a_i \cup 1{:}s_{i+1} \models_{bg}$ ***RESET***, $X^{s_{i+1}} = r_{i+1}$ and $X^{s_i} = r_i$, it is immediate that $\mathsf{Reset}_e(r_i, r_{i+1})$ is true.
- By Lemma 6, (v_i, r_i) and (v_{i+1}, r_{i+1}) are states.

Consequently, we conclude that $(v_i, r_i) \xrightarrow{\sigma_i} (v_{i+1}, r_{i+1})$ is a transition in T_H.

Case 2: $(\mathsf{hevent}(e))^{a_i} = \text{FALSE}$ for all edges $e = (v, v') \in E$. By construction, $\sigma_i = (\textit{Dur})^{a_i}$ where $(\textit{Dur})^{a_i} \in \mathcal{R}_{\geq 0}$. By Lemma 6, (v_i, r_i) and (v_{i+1}, r_{i+1}) are states of T_H. Since $0{:}s_i \cup 0{:}a_i \cup 1{:}s_{i+1} \models$ ***MODE***, it follows that $\textit{Mode}^{s_i} = \textit{Mode}^{s_{i+1}}$. As a result, $v_i = v_{i+1}$. We are to show that there is a differentiable function $f : [0, \sigma_i] \to \mathcal{R}^n$, with the first derivative $\dot{f} : [0, \sigma_i] \to \mathcal{R}^n$ such that: (i) $f(0) = r_i$ and $f(\sigma_i) = r_{i+1}$ and (ii) for all reals $\epsilon \in (0, \sigma_i)$, both $\mathsf{Inv}_{v_i}(f(\epsilon))$ and $\mathsf{Flow}_{v_i}(\dot{f}(\epsilon))$ are true. We now check these conditions for two cases.

1. $\sigma_i = 0$: Since $0{:}s_i \cup 0{:}a_i \cup 1{:}s_{i+1} \models_{bg} (B4)$, it is clear that $r_{i+1} = r_i$. This satisfies condition (i) since $f(\sigma_i) = f(0) = r_{i+1} = r_i$. Condition (ii) is trivially satisfied since there is no $\epsilon \in (0, 0)$.
2. $\sigma_i > 0$: Define $f(t) = r_i + t * (r_{i+1} - r_i)/\sigma_i$. We check that f satisfies the above conditions:
   - $f(t)$ is differentiable over $[0, \sigma_i]$.
   - It is clear that $f(0) = r_i$ and $f(\sigma_i) = r_{i+1}$.
   - We check that for any $\epsilon \in (0, \sigma)$, $\mathsf{Inv}_v(f(\epsilon))$ is true. From $0{:}s_i \cup 1{:}s_{i+1} \models_{bg} (B5)$, it follows that $\mathsf{Inv}_{v_i}(f(0))$ and $\mathsf{Inv}_{v_i}(f(\sigma_i))$ are true. Since the values of X that makes $\mathsf{Inv}_{v_i}(X)$ form a convex region in $\mathcal{R}^n$ and $f(t)$ is a linear function, it follows that for $\epsilon \in (0, \sigma)$, $\mathsf{Inv}_{v_i}(f(\epsilon))$ is true.
   - We check that for any $\epsilon \in (0, \sigma)$, $\mathsf{Flow}_{v_i}(\dot{f}(\epsilon))$ is true. From (B3), it follows that $\mathsf{Flow}_{v_i}((f(\sigma_i) - f(0))/\sigma_i)$ is true. Since $f(t)$ is a linear function, it follows that for any $\epsilon \in (0, \sigma_i)$, $\dot{f}(\epsilon) = (f(\sigma_i) - f(0))/\sigma_i$. As a result, $\mathsf{Flow}_{v_i}(\dot{f}(\epsilon))$ is true

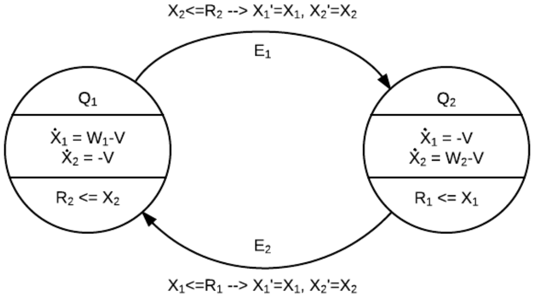
Consequently, we conclude that $(v_i, r_i) \xrightarrow{\sigma_i} (v_{i+1}, r_{i+1})$ is a transition in T_H. □

Lemma 2
q' is a path in the transition system of T_H.

Proof. By Lemma 6, each (v_i, r_i) is a state in T_H. By Lemma 7, each $(v_i, r_i) \xrightarrow{\sigma_i} (v_{i+1}, r_{i+1})$ is a transition in T_H. So q' is a path in T_H. □

Appendix C Examples

C.1 Water Tank Example



This example describes a water tank example with 2 tanks X_1 and X_2. Here R_1 and R_2 are constants that describe the lower bounds of the level of water in the respective tanks. W_1 and W_2 are constants that define the rate at which water is being added to the respective tanks and V is the constant rate at which water is draining from the tanks. We assume that water is added only one tank at a time.

Assuming $W_1 = W_2 = 7.5$, $V = 5$, $R_1 = R_2 = 0$ and initially the level of water in the respective tanks are $X_1 = 0, X_2 = 8$, then the goal is to find a way to add water to each of the tanks with the passage of time.

C.1.1 Hybrid Automata Components

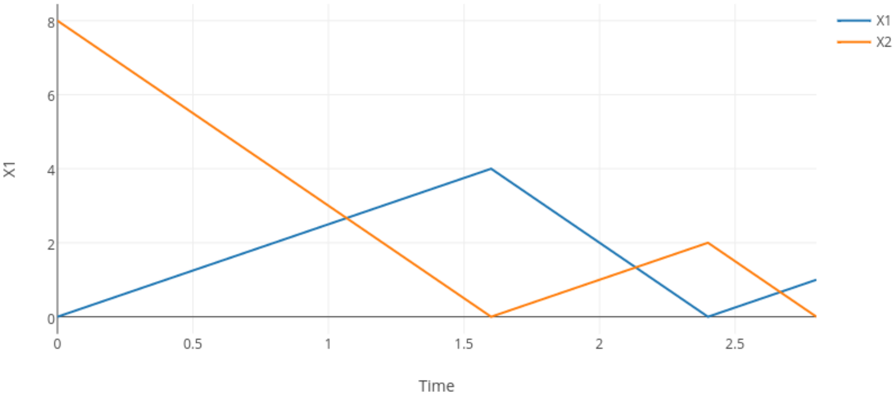
- Variables:
  - — $X_1, X_1', \dot{X}_1$
  - — $X_2, X_2', \dot{X}_2$
- States:
  - — Q_1 (mode=1)
  - — Q_2 (mode=2)
- Directed Graph: The graph is given above
- Invariants:
  - — $\mathsf{Inv}_{Q_1}(X) : X_2 \geq \mathrm{R}_2$
  - — $\mathsf{Inv}_{Q_2}(X) : X_1 \geq \mathrm{R}_1$
- Flow:
  - — $\mathsf{Flow}_{Q_1}(X) : \dot{X}_1 = \mathrm{W}_1 - \mathrm{V} \ \wedge \ \dot{X}_2 = -\mathrm{V}.$
  - — $\mathsf{Flow}_{Q2}(X) : \dot{X}_1 = -\mathrm{V} \ \wedge \ \dot{X}_2 = \mathrm{W}_2 - \mathrm{V}.$
- Guard and Reset:
  - — $\mathsf{Guard}_{(Q_1,Q_2)}(X) : X_2 \leq \mathrm{R}_2.$
  - — $\mathsf{Guard}_{(Q_2,Q_1)}(X) : X_1 \leq \mathrm{R}_1.$
  - — $\mathsf{Reset}_{(Q_1,Q_2)}(X, X') : X_1' = X_1 \ \wedge \ X_2' = X_2.$
  - — $\mathsf{Reset}_{(Q_2,Q_1)}(X, X') : X_1' = X_1 \ \wedge \ X_2' = X_2.$

C.1.2 In the Input Language of CPLUS2ASPMT

```
% File: water.cp

:- constants
x1,x2    :: simpleFluent(real[0..30]);
mode     :: inertialFluent(real[1..2]);
e1,e2    :: exogenousAction;
wait     :: action;
duration :: exogenousAction(real[0..10]).

:- variables
X11,X21,X10,X20,T,X.

exogenous x1.
exogenous x2.

% Guard
nonexecutable e1 if -(x2<=r2).
nonexecutable e2 if -(x1<=r1).

% Reset
constraint (x1=X10 & x2=X20) after x1=X10 & x2=X20 & e1.
constraint (x1=X10 & x2=X20) after x1=X10 & x2=X20 & e2.

% Mode
nonexecutable e1 if -(mode=1).
nonexecutable e2 if -(mode=2).
e1 causes mode=2.
e2 causes mode=1.

% Duration
e1 causes duration=0.
e2 causes duration=0.

% Wait
default wait.
e1 causes ~wait.
e2 causes ~wait.

% Flow
constraint (x1=X11 & x2=X21 ->> (((X11-X10)//T)=w1-v & ((X21-X20)//T)=-v))
    after mode=1 & x1=X10 & x2=X20 & duration=T & wait & T>0.

constraint (x1=X10 & x2=X20)
    after mode=1 & x1=X10 & x2=X20 & duration=0 & wait.

constraint (x1=X11 & x2=X21 ->> (((X11-X10)//T)=-v & ((X21-X20)//T)=w2-v))
    after mode=2 & x1=X10 & x2=X20 & duration=T & wait & T>0.

constraint (x1=X10 & x2=X20)
    after mode=2 & x1=X10 & x2=X20 & duration=0 & wait.

% Invariant
```

```
constraint (mode=1 ->> (x2=X ->> X>=r2)).
constraint (mode=2 ->> (x1=X ->> X>=r1)).

:- query
label :: test;
maxstep :: 6;
0:mode=1;
0:x1 = 0;
0:x2 = 8;
2:mode=2;
4:mode=1;
6:mode=2.
```

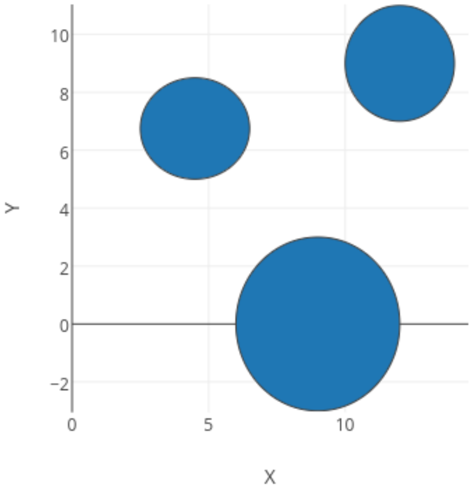
C.1.3 Output



```
Command: cplus2aspmt water.cp -c maxstep=6 -c query=test -c w1=7.5 -c w2=7.5
    -c v=5 -c r1=0 =c r2=0

Solution:
duration_0_ : [ ENTIRE ] = [1.6, 1.6]
duration_1_ : [ ENTIRE ] = [0, 0]
duration_2_ : [ ENTIRE ] = [0.7999999999999998, 0.8000000000000003]
duration_3_ : [ ENTIRE ] = [0, 0]
duration_4_ : [ ENTIRE ] = [0.3999999999999999, 0.4000000000000002]
duration_5_ : [ ENTIRE ] = [0, 0]
mode_0_ : [ ENTIRE ] = [1, 1]
mode_1_ : [ ENTIRE ] = [1, 1]
mode_2_ : [ ENTIRE ] = [2, 2]
mode_3_ : [ ENTIRE ] = [2, 2]
mode_4_ : [ ENTIRE ] = [1, 1]
mode_5_ : [ ENTIRE ] = [1, 1]
mode_6_ : [ ENTIRE ] = [2, 2]
x1_0_ : [ ENTIRE ] = [0, 0]
x1_1_ : [ ENTIRE ] = [4, 4.000000000000001]
```

```
x1_2_ : [ ENTIRE ] = [4, 4.000000000000001]
x1_3_ : [ ENTIRE ] = [0, 0]
x1_4_ : [ ENTIRE ] = [0, 0]
x1_5_ : [ ENTIRE ] = [0.9999999999999998, 1.000000000000001]
x1_6_ : [ ENTIRE ] = [0.9999999999999998, 1.000000000000001]
x2_0_ : [ ENTIRE ] = [8, 8]
x2_1_ : [ ENTIRE ] = [0, 0]
x2_2_ : [ ENTIRE ] = [0, 0]
x2_3_ : [ ENTIRE ] = [2, 2.000000000000001]
x2_4_ : [ ENTIRE ] = [2, 2.000000000000001]
x2_5_ : [ ENTIRE ] = [0, 0]
x2_6_ : [ ENTIRE ] = [0, 0]
true_a : Bool = true
false_a : Bool = false
e1_0_ : Bool = false
e1_1_ : Bool = true
e1_2_ : Bool = false
e1_3_ : Bool = false
e1_4_ : Bool = false
e1_5_ : Bool = true
e2_0_ : Bool = false
e2_1_ : Bool = false
e2_2_ : Bool = false
e2_3_ : Bool = true
e2_4_ : Bool = false
e2_5_ : Bool = false
qlabel_init_ : Bool = true
wait_0_ : Bool = true
wait_1_ : Bool = false
wait_2_ : Bool = true
wait_3_ : Bool = false
wait_4_ : Bool = true
wait_5_ : Bool = false
delta-sat with delta = 0.00100000000000000
Total time in milliseconds: 4328
```

C.2 Turning Car — Non-convex Invariants



Consider a car that is moving at a constant speed of 1 unit. The car is initially at origin where $x = 0$ and $y = 0$ and $\theta = 0$. Additionally there are pillars defined by the equations $(x-6)^2+y^2 \leq 9$,$(x-5)^2+(y-7)^2 \leq 4$,$(x-12)^2+(y-9)^2 \leq 4$. The goal is to find a plan such that the car ends up at $x = 13$ and $y = 0$ without hitting the pillars.

The dynamics of the car is as follows:

- Moving Straight

$$\frac{d[x]}{dt} = \cos(\theta),\ \frac{d[y]}{dt} = \sin(\theta),\ \frac{d[theta]}{dt} = 0$$

- Turning Left

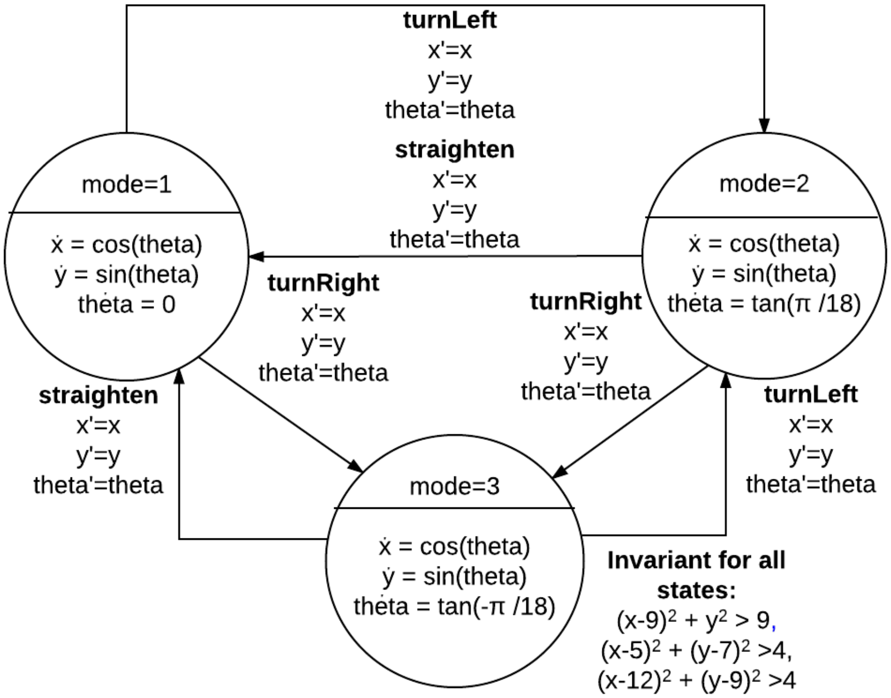
$$\frac{d[x]}{dt} = \cos(\theta),\ \frac{d[y]}{dt} = \sin(\theta),\ \frac{d[theta]}{dt} = \tan(\frac{\pi}{18})$$

- Turning Right

$$\frac{d[x]}{dt} = \cos(\theta),\ \frac{d[y]}{dt} = \sin(\theta),\ \frac{d[theta]}{dt} = \tan(-\frac{\pi}{18})$$

We assume the car is a pint. For the car not to hit the pillars, the invariants are $(x-9)^2+y^2 > 9$,$(x-5)^2+(y-7)^2 > 4$,$(x-12)^2+(y-9)^2 > 4$.

C.2.1 Hybrid Automata Components



- Variables:
  - $X, X', \dot{X}$
  - $Y, Y', \dot{Y}$
  - $Theta, Theta', Th\dot{e}ta$
- States:
  - MoveStraight ($mode = 1$)
  - MoveLeft ($mode = 2$)
  - MoveRight ($mode = 3$)
- Directed Graph: The graph is given above.
- H-events:
  - Straighten
  - TurnLeft
  - TurnRight
- Invariants:
  - $\mathsf{Inv}(allmodes) : ((X-9)^2 + Y^2 > 9) \wedge\ ((X-5)^2 + (Y-7)^2 > 4) \wedge\ ((X-12)^2 + (Y-9)^2 > 4))$
- Flow:
  - $\mathsf{Flow}(1)(X, Y, Theta) : \dot{X} = sin(Theta)\ \wedge\ \dot{Y} = cos(Theta) \wedge\ Th\dot{e}ta = 0.$

- — $\mathsf{Flow}(2)(X, Y, Theta)\ :\ \dot{X} = sin(Theta)\ \wedge\ \dot{Y} = cos(Theta) \wedge\ \dot{Theta} = tan(\pi/18)$.
- — $\mathsf{Flow}(3)(X, Y, Theta)\ :\ \dot{X} = sin(Theta)\ \wedge\ \dot{Y} = cos(Theta) \wedge\ \dot{Theta} = tan(-\pi/18)$.

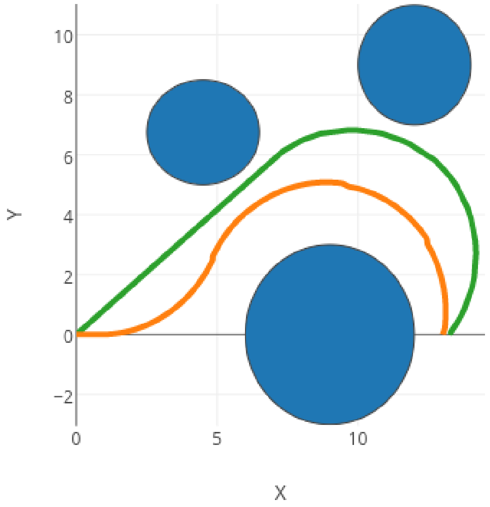
- Reset:
  - — $\mathsf{Reset}((MoveRight, MoveStraight)) : X' = X\ \wedge\ Y' = Y\ \wedge\ Theta' = Theta$
  - — $\mathsf{Reset}((MoveLeft, MoveStraight)) : X' = X\ \wedge\ Y' = Y\ \wedge\ Theta' = Theta$
  - — $\mathsf{Reset}((MoveStraight, MoveLeft)) : X' = X\ \wedge\ Y' = Y\ \wedge\ Theta' = Theta$
  - — $\mathsf{Reset}((MoveRight, MoveLeft)) : X' = X\ \wedge\ Y' = Y\ \wedge\ Theta' = Theta$
  - — $\mathsf{Reset}((MoveStraight, MoveRight)) : X' = X\ \wedge\ Y' = Y\ \wedge\ Theta' = Theta$
  - — $\mathsf{Reset}((MoveLeft, MoveRight)) : X' = X\ \wedge\ Y' = Y\ \wedge\ Theta' = Theta$

C.2.2 In the Input Language of CPLUS2ASPMT

```
% File: car.cp

:- constants
x         :: differentiableFluent(0..40);
y         :: differentiableFluent(-50..50);
theta     :: differentiableFluent(-50..50);
straighten,
turnLeft,
turnRight :: exogenousAction.

:- variables
X,X0,S,Y,X1,X2,D,D1,T,RP,R.

% Reset
constraint (x=D & y=X0 & theta=X1) after x=D & y=X0 & theta=X1 & turnLeft.
constraint (x=D & y=X0 & theta=X1) after x=D & y=X0 & theta=X1 & turnRight.
constraint (x=D & y=X0 & theta=X1) after x=D & y=X0 & theta=X1 & straighten.

% Mode
straighten causes mode=1.
turnLeft causes mode=2.
turnRight causes mode=3.
nonexecutable straighten if mode=1.
nonexecutable turnLeft if mode=2.
nonexecutable turnRight if mode=3.

% Duration
straighten causes duration=0.
turnRight causes duration=0.
turnLeft causes duration=0.

% Wait
default wait.
straighten causes ~wait.
turnLeft causes ~wait.
turnRight causes ~wait.
```

```
% Rates
derivative of theta is 0 if mode=1.
derivative of y is sin(theta) if mode=1.
derivative of x is cos(theta) if mode=1.

derivative of theta is tan(0.226893) if mode=2.
derivative of y is  sin(theta) if mode=2.
derivative of x is cos(theta) if mode=2.

derivative of theta is tan(-0.226893) if mode=3.
derivative of y is sin(theta) if mode=3.
derivative of x is cos(theta) if mode=3.

% Invariant
constraint (x=X & y=Y ->> (X-9)*(X-9) + Y*Y > 9).
always_t (x=X & y=Y ->> (X-9)*(X-9) + Y*Y > 9) if mode=1.
always_t (x=X & y=Y ->> (X-9)*(X-9) + Y*Y > 9) if mode=2.
always_t (x=X & y=Y ->> (X-9)*(X-9) + Y*Y > 9) if mode=3.

constraint (x=X & y=Y ->> (X-5)*(X-5) + (Y-7)*(Y-7)>4).
always_t (x=X & y=Y ->> (X-5)*(X-5) + (Y-7)*(Y-7)>4) if mode=1.
always_t (x=X & y=Y ->> (X-5)*(X-5) + (Y-7)*(Y-7)>4) if mode=2.
always_t (x=X & y=Y ->> (X-5)*(X-5) + (Y-7)*(Y-7)>4) if mode=3.

constraint (x=X & y=Y ->> (X-12)*(X-12) + (Y-9)*(Y-9)>4).
always_t (x=X & y=Y ->> (X-12)*(X-12) + (Y-9)*(Y-9)>4) if mode=1.
always_t (x=X & y=Y ->> (X-12)*(X-12) + (Y-9)*(Y-9)>4) if mode=2.
always_t (x=X & y=Y ->> (X-12)*(X-12) + (Y-9)*(Y-9)>4) if mode=3.

:- query
label :: test;
0:x=0;
0:y=0;
0:theta=0.69183;
0:mode=1;
3:x=13;
3:y=0.
```

C.2.3 Output



```
Command: cplus2aspmt car.cp -c maxstep=3 -c query=test

Output:
Solution:
duration_0_ : [ ENTIRE ] = [8.250457763671875, 8.25128173828125]
duration_1_ : [ ENTIRE ] = [0, 0]
duration_2_ : [ ENTIRE ] = [11.80044126510621, 11.80111503601076]
mode_0_ : [ ENTIRE ] = [1, 1]
mode_1_ : [ ENTIRE ] = [1, 1]
mode_2_ : [ ENTIRE ] = [3, 3]
mode_3_ : [ ENTIRE ] = [3, 3]
theta_0_ : [ ENTIRE ] = [0.6918, 0.6918000000000001]
theta_0_t : [ ENTIRE ] = [0.6918, 0.6918000000000001]
theta_1_t : [ ENTIRE ] = [0.6918, 0.6918000000000001]
theta_2_t : [ ENTIRE ] = [-2.031548557285888, -2.03139307087505]
x_0_ : [ ENTIRE ] = [0, 0]
x_0_t : [ ENTIRE ] = [6.353669267319927, 6.354303809342038]
x_1_t : [ ENTIRE ] = [6.353669267319927, 6.354303809342038]
x_2_t : [ ENTIRE ] = [13, 13]
y_0_ : [ ENTIRE ] = [0, 0]
y_0_t : [ ENTIRE ] = [5.263168261764744, 5.263693895267374]
y_1_t : [ ENTIRE ] = [5.263168261764744, 5.263693895267374]
y_2_t : [ ENTIRE ] = [0, 0]
true_a : Bool = true
false_a : Bool = false
qlabel_test_ : Bool = true
straighten_0_ : Bool = false
straighten_1_ : Bool = false
straighten_2_ : Bool = false
turnLeft_0_ : Bool = false
turnLeft_1_ : Bool = false
turnLeft_2_ : Bool = false
turnRight_0_ : Bool = false
```

```
turnRight_1_ : Bool = true
turnRight_2_ : Bool = false
wait_0_ : Bool = true
wait_1_ : Bool = false
wait_2_ : Bool = true
delta-sat with delta = 0.00100000000000000
Total time in milliseconds: 296721
```

Appendix D Experiments

Steps	1	3	6	8	10
`dReach` (encoding from (Bryce et al. 2015))	0.098	0.225	0.690	2.123	3.143
CPLUS2ASPMT	0.198	7.55	18.23	88.93	> 600

Table D 1. *Runtime Comparison (seconds)*

We compare the run time of the system in (Bryce et al. 2015) and CPLUS2ASPMT for the car domain (Fox and Long 2006) and the result is shown in Table D 1. In (Bryce et al. 2015), the encoding was in the language of `dReach`, which calls `dReal` internally. The computation is optimized for pruning invalid paths of a transition system. It filters out invalid paths using heuristics described in their paper, generates a compact logical encoding, and makes a call to `dReal` to decide reachability properties. On the other hand CPLUS2ASPMT generates a one-time large encoding without filtering paths and calls `dReal` once. From the table we see that the system presented in (Bryce et al. 2015) does perform better than CPLUS2ASPMT. As steps increases the difference in run time also increases.

It may be possible to improve the run time of CPLUS2ASPMT by leveraging incremental answer set computation and path heuristics, which we leave for future work.