Action Language $\mathcal{BC}$+: Preliminary Report

Joseph Babb and Joohyung Lee

School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, USA
{Joseph.Babb, joolee}@asu.edu

Abstract. Recently, action language $\mathcal{BC}$, which combines the attractive features of action languages $\mathcal{B}$ and $\mathcal{C}$+, was proposed. While $\mathcal{BC}$ allows Prolog-style recursive definitions that are not available in $\mathcal{C}$+, it is less expressive than $\mathcal{C}$+ in other ways, such as inability to express non-Boolean and non-exogenous actions. We present a new action language $\mathcal{BC}$+, which encompasses $\mathcal{BC}$ and $\mathcal{C}$+. The syntax of $\mathcal{BC}$+ is identical to the syntax of $\mathcal{C}$+ allowing arbitrary propositional formulas in the causal laws, but its semantics is defined in terms of propositional formulas under the stable model semantics instead of nonmonotonic causal theories. This approach allows many useful ASP constructs, such as choice rules and aggregates, to be directly used in language $\mathcal{BC}$+, and exploits computational methods available in ASP solvers.

1 Introduction

Action languages are formal models of parts of natural language that are used for describing properties of actions. The semantics of such languages are given in terms of transition systems—directed graphs whose vertices represent states and edges represent actions that affect states. These languages are often viewed as high level notations of nonmonotonic logics. For instance, languages $\mathcal{A}$ [Gelfond and Lifschitz, 1993] and $\mathcal{B}$ [Gelfond and Lifschitz, 1998, Section 5] can be translated into logic programs under the stable model semantics; languages $\mathcal{C}$ [Giunchiglia and Lifschitz, 1998] and $\mathcal{C}$+ [Giunchiglia *et al.*, 2004] are defined in terms of nonmonotonic causal theories.

Recently, action language $\mathcal{BC}$ was proposed, which combines the attractive features of $\mathcal{B}$ and $\mathcal{C}$+. Like $\mathcal{C}$+, the inertia assumption is not built into $\mathcal{BC}$, so fluents whose behavior is described by defaults other than inertia can be conveniently represented in $\mathcal{BC}$.[1] Unlike $\mathcal{C}$+, the language allows the user to represent Prolog-style recursive definitions available in $\mathcal{B}$, which are useful in describing complex relations among fluents.[2]

However, $\mathcal{BC}$ is less expressive than $\mathcal{C}$+ in several other ways. It assumes that every action is Boolean and can be executed arbitrarily. The assumption is too strong to describe complex relations among actions, defeasible causal laws [Giunchiglia *et al.*, 2004, Section 4.3], action attributes [Giunchiglia *et al.*, 2004, Section 5.6], and additive fluents [Lee and Lifschitz, 2003] that $\mathcal{C}$+ is able to express conveniently. Also, $\mathcal{C}$+ allows complex formulas in causal laws, but $\mathcal{BC}$ allows only conjunctions of atoms.

[1] See the Leaking Container example [Lee *et al.*, 2013], also reproduced in Section 6.3.

[2] See the definition of *InTower* in the Blocks World from [Lee *et al.*, 2013], also reproduced in Section 5.

In this note, we propose a new action description language $\mathcal{BC}$+. The language is sufficiently expressive to encompass all the features of $\mathcal{BC}$ and the definite fragment of $\mathcal{C}$+. The syntax of this language is essentially identical to $\mathcal{C}$+, but unlike $\mathcal{C}$+, whose semantics was defined via a translation into nonmonotonic causal theories, the semantics of $\mathcal{BC}$+ is defined in terms of the stable model semantics like $\mathcal{BC}$. However, unlike $\mathcal{BC}$, causal laws in $\mathcal{BC}$+ are general enough to allow many useful ASP constructs, such as choice rules and aggregates. The resulting language is very expressive, and it is easy to embed $\mathcal{BC}$ and the definite fragment of $\mathcal{C}$+ as its special cases.

The paper is organized as follows. Sections 2 and 3 introduce the syntax and the semantics of $\mathcal{BC}$+. Section 4 introduces useful abbreviations, which follow the ones available in $\mathcal{C}$+. Section 5 presents an example that illustrates some advantages of $\mathcal{BC}$+. Sections 6 and 7 show how $\mathcal{BC}$ and $\mathcal{C}$+ can be embedded in $\mathcal{BC}$+, and what are the advantages of $\mathcal{BC}$+ over the two languages. Section 8 shows how propositional formulas under the stable model semantics can be viewed as a special case of $\mathcal{BC}$+.

2 Syntax of $\mathcal{BC}$+

The syntax of language $\mathcal{BC}$+ is essentially identical to the syntax of $\mathcal{C}$+.[3] The following largely repeats the syntax description in Section 4.2 from [Giunchiglia *et al.*, 2004]. Language $\mathcal{BC}$+ includes a finite set of symbols of two kinds, *fluent constants* and *action constants*. Fluent constants are further divided into *regular* and *statically determined*. A finite set of cardinality ≥ 2, called the *domain*, is assigned to every constant c and denoted by $Dom(c)$.

We consider (propositional) formulas whose atoms have the form $c = v$, where c is a constant, and v is an element of its domain. If the domain of f is $\{\mathbf{f}, \mathbf{t}\}$ then we say that f is *Boolean*. When c is a Boolean constant, we abbreviate $c = \mathbf{t}$ as c and $c = \mathbf{f}$ as $\sim c$.

A *fluent formula* is a formula such that all constants occurring in it are fluent constants. An *action formula* is a formula that contains at least one action constant and no fluent constants.

A *static law* is an expression of the form

$$\textbf{caused } F \textbf{ if } G \tag{1}$$

where F and G are fluent formulas. An *action dynamic law* is an expression of the form (1) in which F is an action formula and G is a formula. A *fluent dynamic law* is an expression of the form

$$\textbf{caused } F \textbf{ if } G \textbf{ after } H \tag{2}$$

where F and G are fluent formulas and H is a formula, provided that F does not contain statically determined constants. Static laws can be used to talk about causal dependencies between fluents in the same state; action dynamic laws can express causal dependencies between concurrently executed actions; fluent dynamic laws can be used for describing direct effects of actions.

[3] In fact, $\mathcal{C}$+ considers "multi-valued" formulas, an extension of propositional formulas, but this difference is not essential in view of Theorem 1 from [Bartholomew and Lee, 2014].

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 a causal law (1) or (2) is called the *head*. Note that statically determined constants are allowed in the heads of static laws, but not in the heads of dynamic laws. This explains the term "statically determined."

We say that action description D is *definite* if the head of every causal law is either $\bot$, an atom $c = v$, or its choice formula $\{c = v\}$, which stands for the propositional formula $c{=}v \vee \neg(c{=}v)$.

3 Semantics of $\mathcal{BC}$+

For any action description D, we will define a sequence of propositional formulas $PF_0(D), PF_1(D), \ldots$ so that the stable models of $PF_m(D)$ represent paths of length m in the transition system corresponding to D. The signature $\sigma_{D,m}$ of $PF_m(D)$ consists of the pairs $i{:}c$ such that

- $i \in \{0, \ldots, m\}$ and c is a fluent constant of D, and
- $i \in \{0, \ldots, m-1\}$ and c is an action constant of D.

The domain of $i : c$ is the same as the domain of c. By $i : F$ we denote the result of inserting $i :$ in front of every occurrence of every constant in a formula F. The translation $PF_m(D)$ is the conjunction of

- $$i{:}F \ \leftarrow \ i{:}G \tag{3}$$
  for every static law (1) in D and every $i \in \{0, \ldots, m\}$, and (3) for every action dynamic law (1) in D and every $i \in \{0, \ldots, m-1\}$;[4]
- $$i{+}1{:}F \ \leftarrow \ (i{+}1{:}G) \wedge (i{:}H) \tag{4}$$
  for every fluent dynamic law (2) in D and every $i \in \{0, \ldots, m-1\}$;
- $$\{0{:}c{=}v\} \tag{5}$$
  for every regular fluent constant c and every $v \in Dom(c)$;
- $$\bigwedge_{v \neq w \ | \ v,w \in Dom(c)} \neg(i{:}c = v \wedge i{:}c = w); \tag{6}$$
  $$\neg\neg \bigvee_{v \in Dom(c)} i{:}c = v \ , \tag{7}$$
  for every constant $i : c$ in $\sigma_{D,m}$, which represent the uniqueness and existence of values for the constant. The conjunction of formulas (6) and (7) can be succinctly represented using the count aggregate as
  $$\leftarrow \neg \ 1\{i{:}c{=}v_1, \ldots, i{:}c{=}v_m\}1 \tag{8}$$

[4] We identify $F \leftarrow G$ with $G \rightarrow F$.

where $\{v_1, \ldots, v_m\}$ is $Dom(c)$. [5]

As described in [Bartholomew and Lee, 2014], in the presence of (6) and (7), the choice rule $\{i : c = v\}$ can be understood as assigning default value v to constant $i : c$ (read "by default, $i : c$ maps to v"). In the absence of additional information about $i : c$, choosing $i : c = v$ is the only option due to the existence of value constraint (7). But if there is a conflicting information about $i : c$, then not choosing $i : c = v$ is the only option, in view of the uniqueness of value constraint (6). We will see that this reading of choice rules in the presence of the uniqueness and existence of value constraints is very convenient in modeling dynamic systems.

Note how the translation $PF_m(D)$ treats regular and statically determined fluent constants in different ways: formulas (5) are included only when c is regular.

In the example below, the following abbreviations are used. If c is a Boolean action constant, we express that F is an effect of executing c by the fluent dynamic law

$$\textbf{caused } F \textbf{ if } \top \textbf{ after } c,$$

which can be abbreviated as

$$c \textbf{ causes } F.$$

If c is an action constant, the expression

$$\textbf{exogenous } c$$

stands for the action dynamic laws

$$\textbf{caused } \{c{=}v\}$$

for all $v \in Dom(c)$. If c is a regular fluent constant, the expression

$$\textbf{inertial } c \tag{9}$$

stands for the fluent dynamic laws

$$\textbf{caused } \{c{=}v\} \textbf{ after } c{=}v$$

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

The transition system shown in Figure 1 can be described by the following action description *SD*, where p is a Boolean regular fluent constant and a is a Boolean action constant:

$$\begin{array}{l} a \textbf{ causes } p, \\ \textbf{exogenous } a, \\ \textbf{inertial } p. \end{array} \tag{10}$$

The theory $PF_m(SD)$ generated from this action description as defined above is as follows. The first line of (10) is turned into the formulas

$$i{+}1{:}p \ \leftarrow \ i{:}a,$$

[5] Aggregates can be viewed as shorthands for propositional formulas [Ferraris, 2005; Lee and Meng, 2009].

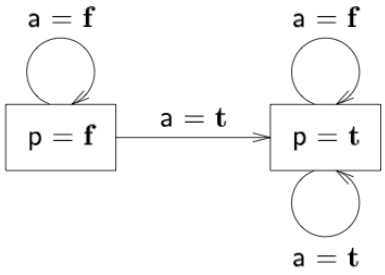

Fig. 1. The transition system described by *SD*.

the second line into

$$\begin{array}{l}\{i{:}a\}\\ \{i{:}{\sim}a\},\end{array} \tag{11}$$

and the third into

$$\begin{array}{l}\{i{+}1{:}p\} \leftarrow i{:}p,\\ \{i{+}1{:}{\sim}p\} \leftarrow i{:}{\sim}p\,.\end{array} \tag{12}$$

In addition,

$$\begin{array}{l}\{0{:}p\},\\ \{0{:}{\sim}p\}\end{array}$$

come from (5), and

$$\begin{array}{l}\leftarrow \neg\, 1\{0{:}p, 0{:}{\sim}p\}1,\\ \leftarrow \neg\, 1\{0{:}a, 0{:}{\sim}a\}1\end{array}$$

come from (8).

The transition system represented by an action description D is defined as follows. A *state* is an interpretation s of σ^{fl} such that $0:s$ is a stable model of $PF_0(D)$. States are the vertices of the transition system represented by D. In view of the existence and uniqueness of value constraints for every state s and every fluent constant c there exists exactly one v such that $c{=}v$ belongs to s; this v is considered the value of c in state s.

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 $0:s \cup 0:e \cup 1:s'$ is a model of $PF_1(D)$. Transitions correspond to the edges of the transition system: for every transition $\langle s, e, s'\rangle$, it contains an edge from s to s' labeled e.

The soundness of this definition is guaranteed by the following fact:

Theorem 1 *For every transition $\langle s, e, s'\rangle$, s and s' are states.*

The stable models of $PF_m(D)$ represent paths of length m in the transition system corresponding to D. For $m = 0$ and $m = 1$, this is clear from the definition of $T(D)$; for $m > 1$ this needs to be verified as the following theorem shows.

For every set X of elements of the signature $\sigma_{D,m}$, let X^i ($i < m$) be the triple consisting of

– the set of atoms A such that $i:A$ belongs to X and A contains fluent constants,

– the set of atoms A such that $i:A$ belongs to X and A contains action constants, and
– the set of atoms A such that $(i+1):A$ belongs to X and A contains fluent constants.

Theorem 2 *For every $m \geq 1$, X is a stable model of $PF_m(D)$ iff $X^0, \ldots, X^{m-1}$ are transitions.*

4 Abbreviations and Restriction on the Syntax of Formulas

Language $\mathcal{BC}$+ allows the same syntax of the abbreviations in $\mathcal{C}$+ (c.f. [Giunchiglia *et al.*, 2004, Appendix B]), but understands them differently. For instance,

– **default** $c=v$ **if** F stands for

$$\textbf{caused } \{c=v\} \textbf{ if } F.$$

This is consistent with the reading of choice formulas in the presence of uniqueness and existence of value constraints as we introduced in Section 3. Similarly,

$$\textbf{default } c=v \textbf{ if } F \textbf{ after } G$$

stands for

$$\textbf{caused } \{c=v\} \textbf{ if } F \textbf{ after } G.$$

This allows us to view the expression (9) as shorthand for

$$\textbf{default } c=v \textbf{ after } c=v.$$

– **constraint** F where F is a fluent formula stands for the static law

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

– **always** F stands for the fluent dynamic law

$$\textbf{caused } \bot \textbf{ if } \top \textbf{ after } \neg F\ ;$$

– **nonexecutable** F **if** G stands for the fluent dynamic law

$$\textbf{caused } \bot \textbf{ if } \top \textbf{ after } F \wedge G.$$

5 Example: Blocks World

An attractive feature of $\mathcal{BC}$+ is that it can use several useful ASP constructs directly in causal laws by resorting to the underlying stable model semantics. For instance, it allows aggregates in causal laws since the semantics of aggregates can be defined in terms of propositional formulas [Ferraris, 2005; Lee and Meng, 2009]. We illustrate this advantage by formalizing the Blocks World domain in $\mathcal{BC}$+.

Let *Blocks* be a finite non-empty set of symbols (block names) that does not include the symbol *Table*. The action description below uses the following fluent and action constants:

– for each $B \in$ *Blocks*, regular fluent constant *Loc*(B) with domain *Blocks* $\cup \{$*Table*$\}$, and statically determined Boolean fluent constant *InTower*(B);
– for each $B \in$ *Blocks* and each $L \in$ *Blocks* $\cup \{$*Table*$\}$, action constant *Move*(B, L).

In the list of static and dynamic laws, B, B_1 and B_2 are arbitrary elements of *Blocks*, and L is an arbitrary element of *Blocks* $\cup \{$*Table*$\}$.

The definition of *InTower*(B):

caused *InTower*(B) **if** *Loc*$(B) =$ *Table*,
caused *InTower*(B) **if** *Loc*$(B) = B_1 \wedge$ *InTower*(B_1),
default $\sim$*InTower*(B).

Blocks don't float in the air:

constraint *InTower*(B).

No two blocks are on the same block:

constraint $\{b :$ *Loc*$(b) = B1\}1$.

Only k towers are allowed to be on the table:

constraint $\{b :$ *Loc*$(b) =$ *Table*$\}k$.

The effect of moving a block:

Move(B, L) **causes** *Loc*$(B) = L$.

A block cannot be moved unless it is clear:

nonexecutable *Move*(B, L) **if** *Loc*$(B_1) = B$.

Concurrent actions are limited by the number g of grippers:

always $\{bl :$ *Move*$(b, l)\}$g.

The commonsense law of inertia:

inertial *Loc*(B).

6 Relation to Language $\mathcal{BC}$

6.1 Review: $\mathcal{BC}$

The signature $\sigma_{D,m}$ for a $\mathcal{BC}$ description D is defined the same as in $\mathcal{BC}$+ except that every action constant is assumed to be Boolean. The main syntactic difference between the causal laws of $\mathcal{BC}$ and the causal laws of $\mathcal{BC}$+ is that the former allows only the conjunction of atoms in the body, and distinguishes between **if** and **if cons** clauses. A $\mathcal{BC}$ *static law* is an expression of the form

$$A_0 \textbf{ if } A_1, \ldots, A_m \textbf{ if cons } A_{m+1}, \ldots, A_n \tag{13}$$

$(n \geq m \geq 0)$, where each A_i is an atom containing a fluent constant. It expresses, informally speaking, that every state satisfies A_0 if it satisfies $A_1, \ldots, A_m$, and $A_{m+1}, \ldots, A_n$ can be consistently assumed.

A $\mathcal{BC}$ *dynamic law* is an expression of the form

$$A_0 \textbf{ after } A_1, \ldots, A_m \textbf{ if cons } A_{m+1}, \ldots, A_n \tag{14}$$

$(n \geq m \geq 0)$, where

- A_0 is an atom containing a regular fluent constant,
- each of $A_1, \ldots, A_m$ is an atom containing a fluent constant, or $a = \mathbf{t}$ where a is an action constant, and
- $A_{m+1}, \ldots, A_n$ are atoms containing fluent constants.

It expresses, informally speaking, that the end state of any transition satisfies A_0 if its beginning state and its action satisfy $A_1, \ldots, A_m$, and $A_{m+1}, \ldots, A_n$ can be consistently assumed about the end state.

An *action description* in language $\mathcal{BC}$ is a finite set consisting of $\mathcal{BC}$ static and $\mathcal{BC}$ dynamic laws.

The semantics of $\mathcal{BC}$ is defined similar to the semantics of $\mathcal{BC}$+ by reduction $PF_m^{\mathcal{BC}}$ to a sequence of logic programs under the stable model semantics. The signature $\sigma_{D,m}$ of $PF_m^{\mathcal{BC}}$ is defined the same as that of PF_m.

For any $\mathcal{BC}$ action description D, by $PF_m^{\mathcal{BC}}(D)$ we denote the conjunction of

- $$i{:}A_0 \ \leftarrow \ i : (A_1 \wedge \cdots \wedge A_m \wedge \neg\neg A_{m+1} \wedge \cdots \wedge \neg\neg A_n) \tag{15}$$
  for every $\mathcal{BC}$ static law (13) in D and every $i \in \{0, \ldots, m\}$;
- $$(i+1){:}A_0 \ \leftarrow \ i{:}(A_1 \wedge \cdots \wedge A_m) \wedge (i{+}1){:}(\neg\neg A_{m+1} \wedge \cdots \wedge \neg\neg A_n) \tag{16}$$
  for every $\mathcal{BC}$ dynamic law (14) in D and every $i \in \{0, \ldots, m-1\}$;
- the formula $i{:}(a = \mathbf{t} \vee a = \mathbf{f})$ for every action constant a and every $i \in \{0, \ldots, m-1\}$;
- the formula (5) for every regular fluent constant c and every element $v \in \mathit{Dom}(c)$;
- the formulas (6) and (7) for every constant $i{:}c$ in $\sigma_{D,m}$.

Note how the translations (15) and (16) treat **if** and **if cons** clauses differently. In $\mathcal{BC}$+, since the formulas are understood under the stable model semantics, there is no need to distinguish between these two clauses.

6.2 Embedding $\mathcal{BC}$ in $\mathcal{BC}$+

Despite the syntactic differences, language $\mathcal{BC}$ can be easily embedded in $\mathcal{BC}$+ as follows. For any $\mathcal{BC}$ description D, we define the translation $\mathit{bc2bcp}(D)$, which turns a $\mathcal{BC}$ description into $\mathcal{BC}$+, as follows:

- replace every causal law (13) with
$$\textbf{caused } A_0 \textbf{ if } A_1 \wedge \cdots \wedge A_m \wedge \neg\neg A_{m+1} \wedge \cdots \wedge \neg\neg A_n; \tag{17}$$

– replace every causal law (14) with

$$\textbf{caused } A_0 \textbf{ if } A_{m+1} \wedge \ldots A_n \textbf{ after } A_0 \wedge \cdots \wedge A_m;$$

– add the causal laws

$$\textbf{caused } \{a = \mathbf{t}\},$$
$$\textbf{caused } \{a = \mathbf{f}\}$$

for every action constant a.

Theorem 3 *For any action description D in language $\mathcal{BC}$, the transition system described by D is identical to the transition system described by bc2bcp(D) in language $\mathcal{BC}$+.*

The proof can be established by showing that the propositional formula $PF_m^{\mathcal{BC}}(D)$ and the propositional formula PF_m(*bc2bcp(D)*) have the same stable models.

6.3 Advantages of $\mathcal{BC}$+ over $\mathcal{BC}$

In $\mathcal{BC}$, every action is assumed to be Boolean and exogenous. This is too strong an assumption that prevents us from describing defeasible causal laws [Giunchiglia *et al.*, 2004, Section 4.3], action attributes [Giunchiglia *et al.*, 2004, Section 5.6], and additive fluents [Lee and Lifschitz, 2003] that $\mathcal{C}$+ is able to express naturally. Also, syntactically, $\mathcal{BC}$ is not expressive enough to describe complex relations among actions. For a simple example, in $\mathcal{BC}$+ we can express that action a_1 is not executable when a_2 is not executed at the same time as

$$\textbf{caused } \bot \textbf{ if } \top \textbf{ after } a_1 \wedge \neg a_2$$

which is not syntactically allowed in $\mathcal{BC}$.

On the other hand, the presence of choice formulas in the head of $\mathcal{BC}$+ causal laws and the different treatment of A and $\neg\neg A$ in the bodies may look subtle to those who are not familiar with the stable model semantics for propositional formulas. Fortunately, in many cases one can avoid choice formulas and double negations by instead using intuitive **default** abbreviations introduced in Section 4. For instance, consider the leaking container example from [Lee *et al.*, 2013] in which a container loses k units of liquid by default. This example illustrates the advantages of $\mathcal{BC}$ over B that is able to express the default other than inertia. In this domain, the fact that *Amount* gets decreased by default is represented in $\mathcal{BC}$+ using the **default** abbreviation:

$$\textbf{default } \mathit{Amount} = x \textbf{ after } \mathit{Amount} = x + k. \tag{18}$$

This abbreviation in $\mathcal{BC}$+ stands for

$$\textbf{caused } \{\mathit{Amount} = x\} \textbf{ after } \mathit{Amount} = x + k,$$

which is further turned into propositional formulas

$$\{i+1 : \mathit{Amount} = x\} \leftarrow i : \mathit{Amount} = x + k \tag{19}$$

($i < m$). On the other hand, the abbreviation (18) in $\mathcal{BC}$ stands for the causal law

$$\textbf{caused } \mathit{Amount} = x \textbf{ after } \mathit{Amount} = x + k \textbf{ if cons } \mathit{Amount} = x,$$

which is further turned into

$$i{+}1{:}\mathit{Amount}{=}x \ \leftarrow i{:}\mathit{Amount}{=}x{+}k \wedge \neg\neg(i{:}\mathit{Amount}{=}x)$$

($i < m$), which is strongly equivalent to (19).

7 Relation to $\mathcal{C}$+

7.1 Review: $\mathcal{C}$+ in ASP

The signature $\sigma_{D,m}$ for a $\mathcal{C}$+ description D is defined the same as in $\mathcal{BC}$+. As mentioned earlier, the syntax of causal laws in $\mathcal{C}$+ is the same as the syntax of causal laws in $\mathcal{BC}$+. That is, A $\mathcal{C}$+ *static law* is an expression of the form

$$\textbf{caused } F \textbf{ if } G \tag{20}$$

where F and G are fluent formulas. A $\mathcal{C}$+ *action dynamic law* is an expression of the form (20) in which F is an action formula and G is a formula. A $\mathcal{C}$+ *fluent dynamic law* is an expression of the form

$$\textbf{caused } F \textbf{ if } G \textbf{ after } H \tag{21}$$

where F and G are fluent formulas and H is a formula, provided that F does not contain statically determined constants. A $\mathcal{C}$+ *causal law* is a static law, an action dynamic law, or a fluent dynamic law. A $\mathcal{C}$+ *action description* is a finite set of $\mathcal{C}$+ causal laws.

We say that $\mathcal{C}$+ action description D is *definite* if the head of every causal law is either $\bot$ or an atom $c{=}v$.

The original semantics of $\mathcal{C}$+ is defined in terms of reduction to nonmonotonic causal theories in [Giunchiglia *et al.*, 2004]. In [Lee, 2012], the semantics of the definite $\mathcal{C}$+ description is alternatively characterized in terms of the stable model semantics for propositional formulas as follows. [6]

For any definite $\mathcal{C}$+ action description D and any nonnegative integer m, the propositional formula $PF_m^{\mathcal{C}+}(D)$ is defined as follows. The signature of $PF_m^{\mathcal{C}+}(D)$ is defined the same as $PF_m(D)$. The translation $PF_m^{\mathcal{C}+}(D)$ is the conjunction of

– $$i{:}F \leftarrow \ \neg\neg\,(i{:}G) \tag{22}$$
for every static law (20) in D and every $i \in \{0,\dots,m\}$, and for every action dynamic law (20) in D and every $i \in \{0,\dots,m-1\}$;

– $$i{+}1{:}F \leftarrow \ \neg\neg(i{+}1{:}G) \wedge (i{:}H) \tag{23}$$
for every fluent dynamic law (21) in D and every $i \in \{0,\dots,m-1\}$;

– the formula (5) for every regular fluent constant c and every $v \in \mathit{Dom}(c)$.

– the formulas (6) and (7) for every constant $i{:}c$ in $\sigma_{D,m}$.

[6] The translation does not work for nondefinite $\mathcal{C}$+ descriptions, due to the different treatments of the heads under nonmonotonic causal theories and under the stable model semantics.

Notation: s, s' ranges over $\{Switch_1, Switch_2\}$; x, y ranges over $\{Up, Down\}$.

Simple fluent constants:	Domains:
$Status(s)$	$\{Up, Down\}$
Action constants:	Domains:
$Flip(s)$	Boolean

Causal laws:

$Flip(s)$ **causes** $Status(s) = x$ **if** $Status(s) = y$	$(x \neq y)$
caused $Status(s) = x$ **if** $Status(s') = y$	$(s \neq s', x \neq y)$
inertial $Status(s)$	
exogenous $Flip(s)$	

Fig. 2. Two Switch

Compare (22) and (23) for $\mathcal{C}$+ with (3) and (4) for $\mathcal{BC}$+. They are very similar except that in (22) and (23), double negations are always introduced when **if** G part is translated. The disadvantage of this translation is that it does not allow us to represent reflexive transitive closures correctly. This is not the case with translations (3) and (4) for $\mathcal{BC}$+, which does not require prepending double negations.

7.2 Embedding Definite $\mathcal{C}$+ in $\mathcal{BC}$+

For any definite $\mathcal{C}$+ description D, we define the translation *cp2bcp*(D), which turns $\mathcal{C}$+ description into $\mathcal{BC}$+, as follows:

– replace every $\mathcal{C}$+ causal law (20) with

$$\textbf{caused } F \textbf{ if } \neg\neg G;$$

– replace every $\mathcal{C}$+ causal law (21) with

$$\textbf{caused } F \textbf{ if } \neg\neg G \textbf{ after } H.$$

The following theorem asserts the correctness of this translation.

Theorem 4 *For any definite action description D in language $\mathcal{C}$+, the transition system described by D is identical to the transition system described by cp2bcp(D) in language $\mathcal{BC}$+.*

Again, the proof can be established by showing that the propositional formula $PF_m^{\mathcal{C}+}(D)$ and the propositional formula $PF_m(cp2bcp(D))$ have the same stable models.

7.3 Advantages of $\mathcal{BC}$+ over $\mathcal{C}$+

Recall that the syntax of $\mathcal{BC}$+ is essentially the same as the syntax of $\mathcal{C}$+, but its semantics is given via the stable model semantics. An advantage of this approach is that

it allows the advances in ASP directly used in the context of $\mathcal{BC}$+. We already saw that being able to use aggregates in $\mathcal{BC}$+ provides a succinct representation of the Blocks World domain.

Another advantage of $\mathcal{BC}$+ is that it avoids some unintuitive behavior of $\mathcal{C}$+ in representing complex relationship involving fluents. Consider two switches which can be flipped but cannot be both up or down at the same time. If one of them is down and the other is up, the direct effect of flipping only one switch is changing the status of that switch, and the indirect effect is changing the status of the other switch. This domain can be represented in $\mathcal{BC}$+ as shown in Figure 2.

The description in $\mathcal{BC}$+ has the following four transitions possible from the initial state where $Switch_1$ is *Down* and $Switch_2$ is *Up*:
$\langle\{St(Sw_1)=Dn, St(Sw_2)=Up\}, \{\sim Flip(Sw_1), \sim Flip(Sw_2)\}, \{St(Sw_1)=Dn, St(Sw_2)=Up\}\rangle$,
$\langle\{St(Sw_1)=Dn, St(Sw_2)=Up\}, \{Flip(Sw_1), \sim Flip(Sw_2)\}, \{St(Sw_1)=Up, St(Sw_2)=Dn\}\rangle$,
$\langle\{St(Sw_1)=Dn, St(Sw_2)=Up\}, \{\sim Flip(Sw_1), Flip(Sw_2)\}, \{St(Sw_1)=Up, St(Sw_2)=Dn\}\rangle$,
$\langle\{St(Sw_1)=Dn, St(Sw_2)=Up\}, \{Flip(Sw_1), Flip(Sw_2)\}, \{St(Sw_1)=Up, St(Sw_2)=Dn\}\rangle$.

The second and the third transitions exhibit the indirect effect of the action *Flip*. If this description is understood in $\mathcal{C}$+, five transitions are possible from the same initial state: in addition to the four transitions above,
$\langle\{St(Sw_1)=Dn, St(Sw_2)=Up\}, \{\sim Flip(Sw_1), \sim Flip(Sw_2)\}, \{St(Sw_1)=Up, St(Sw_2)=Dn\}\rangle$
is also a transition according to the semantics of $\mathcal{C}$+, which is obviously unintuitive.

8 Embedding ASP Programs in $\mathcal{BC}$+

We defined the semantics of $\mathcal{BC}$+ by reducing the language to propositional formulas under the stable model semantics. The reduction in the opposite direction is also possible.

Any propositional formula F under the stable model semantics can be turned into an action description in $\mathcal{BC}$+ by treating every atom of F as a statically determined fluent constant with Boolean values, and rewriting every formula F as the static law

$$\textbf{caused } F$$

and add

$$\textbf{caused } \{c=\mathbf{f}\}$$

for every constant c.

Proposition 1 *The stable models of a propositional formula are exactly the states of the transition system described by the $\mathcal{BC}$+ description obtained by the translation above.*

The proof uses Theorem 7 from [Bartholomew and Lee, 2012].

9 Conclusion

While many action languages are shown to be turned into logic programs under the stable model semantics, language $\mathcal{BC}$+ exploits the generality of the propositional formulas under the stable model semantics. This approach can be further generalized by

using a more general version of the stable model semantics as the target language, such as the first-order stable model semantics [Ferraris *et al.*, 2011], which would yield a proper generalization of $\mathcal{BC}$+ to the first-order level.

Language $\mathcal{BC}$+ is implemented in CPLUS2ASP [Babb and Lee, 2013], which was originally designed to compute the definite fragment of $\mathcal{C}$+ using ASP solvers. As the translation $PF_m^{\mathcal{C}+}(D)$ for $\mathcal{C}$+ is very similar to the translation $PF_m(D)$ for $\mathcal{BC}$+, this extension is straightforward.

A Selected Proofs

Theorem 1 *For every transition $\langle s, e, s' \rangle$, s and s' are states.*

Proof. We will use the following notations: $SD(i)$ is the set of rules (3) in $PF_m(D)$ obtained from static law (1); $AD(i)$ is the set of rules (3) in $PF_m(D)$ obtained from action dynamic law (1); $FD(i)$ is the set of rules (4) in $PF_m(D)$ obtained from fluent dynamic law (2). For the signature σ of D, σ^r is the subset of σ consisting of regular fluent constants; σ^{sd} is the subset of σ consisting of statically determined fluent constants; σ^{fl} is the union of σ^r and σ^{sd}; σ^{act} is the subset of σ consisting of action constants. The conjunction of formulas (6) and (7) for all symbols in a signature δ will be denoted by UEC_δ.

We refer the notion of SM in [Ferraris *et al.*, 2011].

Since $\langle s, e, s' \rangle$ is a transition, by definition,

$$0{:}s \cup 0{:}e \cup 1{:}s' \models \text{SM}[SD(0) \cup AD(0) \cup FD(0) \cup SD(1);\ 0{:}\sigma^{sd} \cup 0{:}\sigma^{act} \cup 1{:}\sigma^r \cup 1{:}\sigma^{sd}] \\ \wedge UEC_{0:\sigma^{fl} \cup 0:\sigma^{act} \cup 1:\sigma^{fl}}].$$

By the splitting theorem [Ferraris *et al.*, 2009], it follows that

$$0{:}s \cup 0{:}e \cup 1{:}s' \models \text{SM}[SD(0);\ 0{:}\sigma^{sd}]; \tag{24}$$

$$0{:}s \cup 0{:}e \cup 1{:}s' \models \text{SM}[FD(0) \wedge SD(1);\ 1{:}\sigma^r \cup 1{:}\sigma^{sd}]; \tag{25}$$

$$0{:}s \cup 0{:}e \cup 1{:}s' \models UEC_{0:\sigma^{fl} \cup 0:\sigma^{act} \cup 1:\sigma^{fl}}.$$

From (24), we have $0 : s \models \text{SM}[SD(0); 0 : \sigma^{sd}]$, and consequently, $0 : s \models \text{SM}[SD(0) \wedge UEC_{0:\sigma^{fl}}]$, so s is a state.

From (25), by Theorem 2 from [Ferraris *et al.*, 2011], it follows that

$$0{:}s \cup 0{:}e \cup 1{:}s' \models \text{SM}[FD(0) \wedge SD(1);\ 1{:}\sigma^{sd}]. \tag{26}$$

Since $FD(0)$ is negative on $1{:}\sigma^{sd}$, (26) is equivalent to

$$0{:}s \cup 0{:}e \cup 1{:}s' \models \text{SM}[SD(1);\ 1{:}\sigma^{sd}] \wedge FD(0),$$

Consequently, we get

$$1{:}s' \models \text{SM}[SD(1) \wedge UEC_{1:\sigma^{fl}};\ 1{:}\sigma^{sd}],$$

which can be rewritten as

$$0{:}s' \models \text{SM}[SD(0) \wedge UEC_{0:\sigma^{fl}};\ 0{:}\sigma^{sd}],$$

so s' is a state. ∎

Theorem 2 *For every $m \geq 1$, X is a stable model of $PF_m(D)$ iff $X^0, \ldots, X^{m-1}$ are transitions.*

Proof. When $m = 1$, the claim is immediate from the definition of a transition.

I.H. Assume that Y is a stable model of $PF_m(D)$ iff $Y^0, \ldots, Y^{m-1}$ are transitions ($m \geq 1$).

X is a stable model of $PF_{m+1}(D)$ iff X satisfies $UEC_{\sigma(D,m)}$ and

$$\begin{array}{l}\text{SM}[SD(0) \wedge AD(0) \wedge FD(0) \wedge SD(1) \\ \quad \wedge AD(1) \wedge FD(1) \wedge SD(2) \\ \quad \wedge \ \ldots \\ \quad \wedge AD(m{-}1) \wedge FD(m{-}1) \wedge SD(m) \\ \quad \wedge AD(m) \wedge FD(m) \wedge SD(m{+}1)\ ; \\ \qquad 0{:}\sigma^{sd} \cup 0{:}\sigma^{act} \cup 1{:}\sigma^{r} \cup 1{:}\sigma^{sd} \cup 1{:}\sigma^{act} \cup 2{:}\sigma^{r} \cup 2{:}\sigma^{sd} \\ \qquad \cup \cdots \cup (m{-}1){:}\sigma^{act} \cup m{:}\sigma^{r} \cup m{:}\sigma^{sd} \cup m{:}\sigma^{act} \cup (m{+}1){:}\sigma^{r} \wedge (m{+}1){:}\sigma^{sd}].\end{array} \tag{27}$$

By the splitting theorem [Ferraris *et al.*, 2009], the fact that X satisfies (27) is equivalent to saying that X satisfies

$$\begin{array}{l}\text{SM}[SD(0) \wedge AD(0) \wedge FD(0) \wedge SD(1) \\ \qquad \wedge AD(1) \wedge FD(1) \wedge SD(2) \\ \qquad \wedge \ \ldots \\ \qquad \wedge AD(m{-}1) \wedge FD(m{-}1) \wedge SD(m)\ ; \\ \qquad\quad 0{:}\sigma^{sd} \cup 0{:}\sigma^{act} \cup 1{:}\sigma^{r} \cup 1{:}\sigma^{sd} \cup 1{:}\sigma^{act} \cup 2{:}\sigma^{r} \cup 2{:}\sigma^{sd} \\ \qquad\quad \cup \cdots \cup (m{-}1){:}\sigma^{act} \cup m{:}\sigma^{r} \cup m{:}\sigma^{sd}]\end{array} \tag{28}$$

and X satisfies

$$\text{SM}[AD(m) \wedge FD(m) \wedge SD(m+1);\ m{:}\sigma^{act} \cup (m{+}1){:}\sigma^{r} \cup (m{+}1){:}\sigma^{sd}]. \tag{29}$$

The fact that $X \models$ (28) is equivalent to saying that

$$X \cap \{0{:}\sigma \cup \cdots \cup m{:}\sigma\} \models (28).$$

By I.H., the latter is equivalent to saying that $X^0, \ldots, X^{m-1}$ are transitions.

Observe that, using the splitting theorem, (28) entails

$$\text{SM}[FD(m{-}1) \wedge SD(m);\ m{:}\sigma^{r} \cup m{:}\sigma^{sd}]. \tag{30}$$

By Theorem 2 from [Ferraris *et al.*, 2011], (30) entails

$$\text{SM}[FD(m{-}1) \wedge SD(m);\ m{:}\sigma^{sd}]. \tag{31}$$

Since $FD(m{-}1)$ is negative on $m{:}\sigma^{sd}$, (31) entails

$$\text{SM}[SD(m);\ m{:}\sigma^{sd}]. \tag{32}$$

By the splitting theorem on (29) and (32), we get

$$\begin{array}{rl}X \models \text{SM}[SD(m) \wedge AD(m) \wedge FD(m) \wedge SD(m+1); & \\ m{:}\sigma^{sd} \cup m{:}\sigma^{act} \cup (m{+}1){:}\sigma^{r} \cup (m{+}1){:}\sigma^{sd}], & \end{array}$$

or equivalently,

$$X \models \text{SM}[\mathit{SD}(0) \wedge \mathit{AD}(0) \wedge \mathit{FD}(0) \wedge \mathit{SD}(1);\ 0{:}\sigma^{sd} \cup 0{:}\sigma^{act} \cup 1{:}\sigma^{r} \cup 1{:}\sigma^{sd}].$$

The latter, together with the fact that X satisfies $\mathit{UEC}_{\sigma(D,1)}$, means that X^m is a transition. ∎

References

[Babb and Lee, 2013] Joseph Babb and Joohyung Lee. Cplus2asp: Computing action language C+ in answer set programming. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 122–134, 2013.

[Bartholomew and Lee, 2012] Michael Bartholomew and Joohyung Lee. Stable models of formulas with intensional functions. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 2–12, 2012.

[Bartholomew and Lee, 2014] Michael Bartholomew and Joohyung Lee. Stable models of multi-valued formulas: Partial vs. total functions. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2014. To appear.

[Ferraris *et al.*, 2009] Paolo Ferraris, Joohyung Lee, Vladimir Lifschitz, and Ravi Palla. Symmetric splitting in the general theory of stable models. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 797–803. AAAI Press, 2009.

[Ferraris *et al.*, 2011] Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. Stable models and circumscription. *Artificial Intelligence*, 175:236–263, 2011.

[Ferraris, 2005] Paolo Ferraris. Answer sets for propositional theories. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 119–131, 2005.

[Gelfond and Lifschitz, 1993] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–322, 1993.

[Gelfond and Lifschitz, 1998] Michael Gelfond and Vladimir Lifschitz. Action languages[7]. *Electronic Transactions on Artificial Intelligence*, 3:195–210, 1998.

[Giunchiglia and Lifschitz, 1998] Enrico Giunchiglia and Vladimir Lifschitz. An action language based on causal explanation: Preliminary report. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 623–630. AAAI Press, 1998.

[Giunchiglia *et al.*, 2004] Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153(1–2):49–104, 2004.

[Lee and Lifschitz, 2003] Joohyung Lee and Vladimir Lifschitz. Describing additive fluents in action language C+. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1079–1084, 2003.

[Lee and Meng, 2009] Joohyung Lee and Yunsong Meng. On reductive semantics of aggregates in answer set programming. In *Procedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 182–195, 2009.

[Lee *et al.*, 2013] Joohyung Lee, Vladimir Lifschitz, and Fangkai Yang. Action language BC: Preliminary report. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[Lee, 2012] Joohyung Lee. Reformulating action language C+ in answer set programming. In Esra Erdem, Joohyung Lee, Yuliya Lierler, and David Pearce, editors, *Correct Reasoning*, volume 7265 of *Lecture Notes in Computer Science*, pages 405–421. Springer, 2012.

[7] `http://www.ep.liu.se/ea/cis/1998/016/`