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

Joseph Babb and **Joohyung Lee**
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Tempe, AZ, 85287, USA
{Joseph.Babb,joolee}@asu.edu

Abstract

Action languages are formal models of parts of natural language that are designed to describe effects of actions. Many of these languages can be viewed as high level notations of answer set programs structured to represent transition systems. However, the form of answer set programs considered in the earlier work is quite limited in comparison with the modern Answer Set Programming (ASP) language, which allows several useful constructs for knowledge representation, such as choice rules, aggregates, and abstract constraint atoms. We propose a new action language called $\mathcal{BC}$+, which closes the gap between action languages and the modern ASP language. Language $\mathcal{BC}$+ is defined as a high level notation of propositional formulas under the stable model semantics. Due to the generality of the underlying language, $\mathcal{BC}$+ is expressive enough to encompass many modern ASP language constructs and the best features of several other action languages, such as $\mathcal{B}$, $\mathcal{C}$, $\mathcal{C}$+ and $\mathcal{BC}$. Computational methods available in ASP solvers are readily applicable to compute $\mathcal{BC}$+, which led us to implement the language by extending system CPLUS2ASP.

Introduction

Action languages are formal models of parts of natural language that are used for describing properties of actions. The semantics of action languages describe transition systems—directed graphs whose vertices represent states and whose edges represent actions that affect the states. Many action languages, such as languages $\mathcal{A}$ (Gelfond and Lifschitz 1993) and $\mathcal{B}$ (Gelfond and Lifschitz 1998, Section 5), can be viewed as high level notations of answer set programs structured to represent transition systems. Languages $\mathcal{C}$ (Giunchiglia and Lifschitz 1998) and $\mathcal{C}$+ (Giunchiglia et al. 2004) are originally defined in terms of nonmonotonic causal theories, but their "definite" fragments can be equivalently turned into logic programs as well, leading to the implementation CPLUS2ASP, which uses ASP solvers for computation (Babb and Lee 2013).

The main advantage of using action languages over ASP programs is their structured abstract representations for describing transition systems, which allows their users to focus on high level descriptions and avoids the "cryptic" syntax and the recurring pattern of ASP rules for representing transition systems. However, the existing work on action languages have two limitations. First, they do not allow many useful ASP constructs, such as choice rules, aggregates, abstract constraint atoms, and external atoms, that have recently been introduced into ASP and have been shown to be very useful. The inability to use these modern constructs in action languages is what often prevents the users from writing in action languages, and instead forces them to write in the language of ASP directly.

Another issue is that even when we do not use such constructs, there are certain limitations that each action language has in comparison with another. For instance, in language $\mathcal{B}$, the frame problem is solved by enforcing in the semantics that every fluent be governed by the commonsense law of inertia, which makes it difficult to represent fluents whose behavior is described by defaults other than inertia. Languages $\mathcal{C}$ and $\mathcal{C}$+ do not have this limitation, but instead they do not handle Prolog-style recursive definitions, such as transitive closure, available in $\mathcal{B}$. The recently proposed language $\mathcal{BC}$ (Lee, Lifschitz, and Yang 2013) combines the attractive features of $\mathcal{B}$ and $\mathcal{C}$+, but it is not a proper generalization. In comparison with $\mathcal{C}$+, it does not allow us to describe complex dependencies among actions, thus it is unable to describe several concepts that $\mathcal{C}$+ is able to express naturally, such as defeasible causal laws (Giunchiglia et al. 2004, Section 4.3) and action attributes (Giunchiglia et al. 2004, Section 5.6).

We present a simple solution to these problems. The main idea is to define an action language in terms of a general stable model semantics, which has not been considered in the work of action languages. We present a new action language called $\mathcal{BC}$+, which is defined as a high level notation of propositional formulas under the stable model semantics in (Ferraris 2005). It has been well studied in ASP that several useful constructs, such as aggregates, abstract constraint atoms, and conditional literals, can be identified with abbreviations of propositional formulas (e.g.,(Ferraris 2005; Pelov, Denecker, and Bruynooghe 2003; Son and Pontelli 2007; Harrison, Lifschitz, and Yang 2014)). Thus, $\mathcal{BC}$+ employs such constructs as well. Further, it is more expressive than the other action languages mentioned above, allowing them to be easily embedded. The computation of $\mathcal{BC}$+ is carried out by ASP solvers, which is implemented by modify-

ing system CPLUS2ASP (Babb and Lee 2013), which was originally designed to compute $\mathcal{C}$+ using ASP solvers.

The paper is organized as follows. We first review propositional formulas under the stable model semantics, and how they express the concept of defaults in the presence of uniqueness and existence of value constraints. Next we present the syntax and the semantics of $\mathcal{BC}$+, as well as a useful fragment and abbreviations. We also show how propositional formulas under the stable model semantics can be viewed as a special case of $\mathcal{BC}$+. Then we show how each of $\mathcal{BC}$ and $\mathcal{C}$+ can be embedded in $\mathcal{BC}$+, and explain the advantages of $\mathcal{BC}$+ over the two languages.

Preliminaries

According to (Ferraris 2005), stable models of a propositional formula are defined as follows. The *reduct* F^X of a propositional formula F relative to a set X of atoms is the formula obtained from F by replacing every maximal subformula that is not satisfied by X with $\bot$. Set X is called a *stable model* of F if X is a minimal set of atoms satisfying F^X. It has been shown that logic programs can be identified with propositional formulas under the stable model semantics in the form of conjunctions of implications.

Throughout this paper, we consider propositional formulas whose signature σ consists of atoms of the form $c=v$, [1] where c is called a *constant* and is associated with a finite set $Dom(c)$ of cardinality ≥ 2, called the *domain*, and v is an element of its domain. If the domain of c is $\{\mathbf{f},\mathbf{t}\}$ then we say that c is *Boolean*; we abbreviate $c=\mathbf{t}$ as c and $c=\mathbf{f}$ as $\sim c$.

Bartholomew and Lee [2014] show that this form of propositional formulas is useful to express the concept of default values on multi-valued fluents. By UEC_σ ("Uniqueness and Existence Constraint") we denote the conjunction of

$$\bigwedge_{v\neq w\,:\,v,w\in Dom(c)} \neg(c=v\wedge c=w), \tag{1}$$

and

$$\neg\neg \bigvee_{v\in Dom(c)} c=v\,. \tag{2}$$

for all constants c of σ. It is clear that an interpretation of σ that satisfies UEC_σ can be identified with a function that maps each constant c into an element in its domain.

Example 1 *Consider a signature σ to be $\{c=1,\ c=2,\ c=3\}$, where c is a constant and $Dom(c)=\{1,2,3\}$. UEC_σ is*

$$\neg(c=1\wedge c=2)\wedge\neg(c=2\wedge c=3)\wedge\neg(c=1\wedge c=3)$$
$$\wedge\,\neg\neg(c=1\vee c=2\vee c=3).$$

Let F_1 be $(c=1\vee\neg(c=1))\wedge UEC_\sigma$. Due to UEC_σ, each of $\{c=1\}$, $\{c=2\}$, and $\{c=3\}$ is a model of F_1, but $\{c=1\}$ is the only stable model of F_1.[2] *This is because the reduct $F_1^{\{c=1\}}$ is equivalent to $c=1$, for which $\{c=1\}$ is the minimal model. On the other hand, for instance, the reduct $F_1^{\{c=2\}}$ is equivalent to $\top$, for which the minimal model is $\emptyset$, not $\{c=2\}$.*

Let F_2 be F_1 conjoined with $c=2$. Similarly, we can check that the only stable model of F_2 is $\{c=2\}$. This illustrates the nonmonotonicity of the semantics.

Note that the presence of double negations is essential in (2). Without them, F_1 would have three stable models: $\{c=1\}$, $\{c=2\}$, and $\{c=3\}$.

In ASP, formulas of the form $F\vee\neg F$ are called *choice formulas* and denoted by $\{F\}^{\mathrm{ch}}$. For example, F_1 above can be written as $\{c=1\}^{\mathrm{ch}}\wedge UEC_\sigma$. As shown in Example 1, in the presence of UEC_σ, a formula of the form $\{c=v\}^{\mathrm{ch}}$ expresses that c has the value v by default, which can be overridden in the presence of other evidence (Bartholomew and Lee 2014).

Aggregates in ASP can be understood as shorthand for propositional formulas as shown in (Ferraris 2005). For instance, the count aggregate expresion $2\{p,q,r\}$ is shorthand for $(p\wedge q)\vee(q\wedge r)\vee(p\wedge r)$.

Syntax of $\mathcal{BC}$+

The syntax of language $\mathcal{BC}$+ is similar to the syntax of $\mathcal{C}$+. The following description repeats the syntax description in Section 4.2 from (Giunchiglia et al. 2004). In language $\mathcal{BC}$+, constants are divided into two groups: *fluent constants* and *action constants*. Fluent constants are further divided into *regular* and *statically determined*.

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

$$\textbf{caused } F \textbf{ if } G \tag{3}$$

where F and G are fluent formulas. An *action dynamic law* is an expression of the form (3) in which F is an action formula and G is a formula. A *fluent dynamic law* is an expression of the form

$$\textbf{caused } F \textbf{ if } G \textbf{ after } H \tag{4}$$

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 be used to express causal dependencies between concurrently executed actions; fluent dynamic laws can be used for describing direct effects of actions.

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 each of causal laws (3) and (4) is called the *head* of the causal law.

Semantics of $\mathcal{BC}$+

For any action description D, we 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 $PF_m(D)$ has the constants $i:c$ such that

[1] Note that $c=v$ is an atom in the propositional signature, and not an equality in first-order logic.

[2] We identify an interpretation with the set of atoms that are true in it.

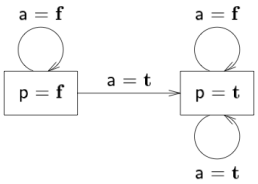

Figure 1: The transition system described by *SD*.

- $i \in \{0,\dots,m\}$ and c is a fluent constant of D, and
- $i \in \{0,\dots,m-1\}$ and c is a 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 formula F. The translation $PF_m(D)$ is the conjunction of

- $$i\!:\!F \ \leftarrow \ i\!:\!G \tag{5}$$
  for every static law (3) in D and every $i \in \{0,\dots,m\}$, and (5) for every action dynamic law (3) in D and every $i \in \{0,\dots,m-1\}$;[3]
- $$i{+}1\!:\!F \ \leftarrow \ (i{+}1\!:\!G) \wedge (i\!:\!H) \tag{6}$$
  for every fluent dynamic law (4) in D and every $i \in \{0,\dots,m-1\}$;
- $$\{0\!:\!c{=}v\}^{\mathrm{ch}} \tag{7}$$
  for every regular fluent constant c and every $v \in \mathit{Dom}(c)$;
- $\mathit{UEC}_{\sigma_{D,m}}$, which can also be abbreviated using the count aggregate as the conjunction of
  $$\leftarrow \neg\, 1\{i\!:\!c{=}v_1,\dots,i\!:\!c{=}v_m\}1 \tag{8}$$
  for every $i\!:\!c$ in $\sigma_{D,m}$ and every $v_1,\dots,v_m$ in $\mathit{Dom}(i\!:\!c)$.

Note how the translation $PF_m(D)$ treats regular and statically determined fluent constants in different ways: formulas (7) are included only when c is regular. Statically determined fluents are useful for describing defined fluents, whose value is determined by the fluents in the same state only. We refer the reader to (Giunchiglia et al. 2004, Section 5) for more details.

As an example, 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.

$$\begin{array}{l}\textbf{caused } p \textbf{ if } \top \textbf{ after } a,\\ \textbf{caused } \{a\}^{\mathrm{ch}},\\ \textbf{caused } \{{\sim}a\}^{\mathrm{ch}},\\ \textbf{caused } \{p\}^{\mathrm{ch}} \textbf{ if } \top \textbf{ after } p,\\ \textbf{caused } \{{\sim}p\}^{\mathrm{ch}} \textbf{ if } \top \textbf{ after } {\sim}p,\end{array} \tag{9}$$

The translation $PF_m(SD)$ turns this description into the following propositional formulas. The first line of (9) is turned into the formulas

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

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

$(0 \le i < m)$, the second and the third lines into

$$\{i\!:\!a\}^{\mathrm{ch}}, \qquad \{i\!:\!{\sim}a\}^{\mathrm{ch}} \tag{10}$$

$(0 \le i < m)$, and the fourth and the fifth lines into

$$\{i{+}1\!:\!p\}^{\mathrm{ch}} \ \leftarrow \ i\!:\!p, \qquad \{i{+}1\!:\!{\sim}p\}^{\mathrm{ch}} \ \leftarrow \ i\!:\!{\sim}p \tag{11}$$

$(0 \le i < m)$. In addition,

$$\{0\!:\!p\}^{\mathrm{ch}}, \qquad \{0\!:\!{\sim}p\}^{\mathrm{ch}}$$

come from (7), and

$$\begin{array}{ll}\leftarrow \neg\, 1\{i\!:\!p,\ i\!:\!{\sim}p\}1 & (0 \le i \le m),\\ \leftarrow \neg\, 1\{i\!:\!a,\ i\!:\!{\sim}a\}1 & (0 \le i < m)\end{array}$$

come from (8).

For every stable model X of $PF_0(D)$, the set of fluent formulas $c{=}v$ such that $0 : c{=}v$ belongs to X is a *state* of D. In view of the uniqueness and existence of value constraints, for every state s and every fluent constant c, there is exactly one v such that $c{=}v$ belongs to s.

For every stable model X of $PF_1(D)$, D includes the transition $\langle s_0, e, s_1\rangle$, where s_i $(i = 0, 1)$ is the set of fluent formulas $c{=}v$ such that $i : c{=}v$ belongs to X, and e is the set of action formulas $c{=}v$ such that $0 : c{=}v$ belongs to X.

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

Theorem 1 *For every transition $\langle s_0, e, s_1\rangle$, s_0 and s_1 are states.*

The stable models of $PF_m(D)$ represent the paths of length m in the transition system corresponding to D. For $m = 0$ and $m = 1$, this is clear from the definition of a transition system; 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 consisting of atoms A such that $i\!:\!A$ belongs to X, and A contains fluent constants,
- the set consisting of atoms A such that $i\!:\!A$ belongs to X, and A contains action constants, and
- the set consisting of atoms A such that $(i+1)\!:\!A$ belongs to X, and A contains fluent constants.

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

For example, $\{0 : {\sim}p, 0 : {\sim}a, 1 : {\sim}p, 1 : a, 2 : p\}$ is a stable model of $PF_2(SD)$, and each of $\langle {\sim}p, {\sim}a, {\sim}p\rangle$ and $\langle {\sim}p, a, p\rangle$ is a transition of *SD*.

Useful Abbreviations

Like $\mathcal{C}$+, several intuitive abbreviations of causal laws can be defined for $\mathcal{BC}$+. Expression "**default** $c{=}v$ **if** F" stands for "**caused** $\{c{=}v\}^{\mathrm{ch}}$ **if** F."[4] This abbreviation is intuitive in view of the reading of choice formulas in the presence of uniqueness and existence of value constraints we introduced earlier (recall Example 1). Similarly,

default $c{=}v$ **if** F **after** G

[4]Here and after, we often omit **if** F if F is $\top$.

stands for "**caused** $\{c=v\}^{\mathrm{ch}}$ **if** F **after** G."

Other abbreviations of $\mathcal{BC}$+ causal laws are defined similarly as in $\mathcal{C}$+.

- If c is a Boolean action constant, we express that F is an effect of executing c by "c **causes** F," which stands for the fluent dynamic law "**caused** F **if** $\top$ **after** c."
- If c is an action constant, the expression "**exogenous** c" stands for the action dynamic laws "**default** $c=v$" for all $v \in \mathit{Dom}(c)$.
- If c is a regular fluent constant, the expression "**inertial** c" stands for the fluent dynamic laws "**default** $c=v$ **after** $c=v$" for all $v \in \mathit{Dom}(c)$.
- "**constraint** F" where F is a fluent formula stands for the static law "**caused** $\bot$ **if** $\neg F$."
- "**always** F" stands for the fluent dynamic law "**caused** $\bot$ **if** $\top$ **after** $\neg F$."
- "**nonexecutable** F **if** G" stands for the fluent dynamic law "**caused** $\bot$ **if** $\top$ **after** $F \wedge G$."

Embedding Formulas under SM 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, rewriting F as the static law "**caused** F," and adding "**default** $c=\mathbf{f}$" for every constant c.

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

It is known that the problem of determining the existence of stable models of propositional formulas is Σ_2^P-complete (Ferraris 2005). The same complexity applies to $\mathcal{BC}$+ in view of Proposition 1. On the other hand, from the translation $PF_m(D)$, a useful fragment in NP can be defined based on the known results in ASP. The following is an instance.

We say that action description D is *definite* if the head of every causal law is either $\bot$, an atom $c=v$, or a choice formula $\{c=v\}^{\mathrm{ch}}$. We say that a formula is a *simple* conjunction if it is a conjunction of atoms and count aggregates, each of which may be preceded by negation.

A *simple* action description is a definite action description such that G in every causal law (3) is a simple conjunction, and G and H in every causal law (4) are simple conjunctions. The next section presents an example of a simple action description.

Example: Blocks World

An attractive feature of $\mathcal{BC}$+ is that aggregates (or more generally, abstract constraints) are directly usable in causal laws because they are understood as abbreviations of propositional formulas (Ferraris 2005; Pelov, Denecker, and Bruynooghe 2003; Son and Pontelli 2007; Lee and Meng 2009). We illustrate this advantage by formalizing an elaboration of the Blocks World from (Lee, Lifschitz, and Yang 2013) by a simple $\mathcal{BC}$+ description.

Let *Blocks* be a non-empty finite 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 $B \in \mathit{Blocks}$, regular fluent constant $\mathit{Loc}(B)$ ("Location") with the domain $\mathit{Blocks} \cup \{\mathit{Table}\}$, and statically determined Boolean fluent constant $\mathit{InTower}(B)$;
- for each $B \in \mathit{Blocks}$, Boolean action constant $\mathit{Move}(B)$;
- for each $B \in \mathit{Blocks}$, action constant $\mathit{Dest}(B)$ ("Destination") with the domain $\mathit{Blocks} \cup \{\mathit{Table}\} \cup \{\mathit{None}\}$, where *None* is an auxiliary symbol for an "undefined" value.

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

Blocks are not on itself:

$$\textbf{constraint } \mathit{Loc}(B) \neq B.$$

The definition of $\mathit{InTower}(B)$:

$$\begin{array}{l}\textbf{caused } \mathit{InTower}(B) \textbf{ if } \mathit{Loc}(B) = \mathit{Table},\\ \textbf{caused } \mathit{InTower}(B) \textbf{ if } \mathit{Loc}(B) = B_1 \wedge \mathit{InTower}(B_1),\\ \textbf{default } \sim\mathit{InTower}(B).\end{array}$$

Blocks don't float in the air:

$$\textbf{constraint } \mathit{InTower}(B).$$

No two blocks are on the same block:

$$\textbf{constraint } \{b : \mathit{Loc}(b) = B\}1.$$

Only k towers are allowed to be on the table (k is a positive integer):

$$\textbf{constraint } \{b : \mathit{Loc}(b) = \mathit{Table}\}\mathrm{k}.$$

The effect of moving a block:

$$\mathit{Move}(B) \textbf{ causes } \mathit{Loc}(B) = L \textbf{ if } \mathit{Dest}(B) = L$$

A block cannot be moved unless it is clear:

$$\textbf{nonexecutable } \mathit{Move}(B) \textbf{ if } \mathit{Loc}(B_1) = B.$$

Concurrent actions are limited by the number g of grippers:

$$\textbf{always } \{b : \mathit{Move}(b)\}\mathrm{g}.$$

The commonsense law of inertia:

$$\textbf{inertial } \mathit{Loc}(B).$$

Actions are exogenous:

$$\begin{array}{c}\textbf{exogenous } \mathit{Move}(B),\\ \textbf{exogenous } \mathit{Dest}(B).\end{array}$$

Dest is an attribute of *Move*:

$$\textbf{always } \mathit{Dest}(B) = \mathit{None} \leftrightarrow \neg\mathit{Move}(B).$$

Besides the inability to represent aggregates, other action languages have difficulties in representing this example. Languages $\mathcal{C}$ and $\mathcal{C}$+ do not allow us to represent the recursive definition of *InTower*. Languages $\mathcal{B}$ and $\mathcal{BC}$ do not allow us to represent action attributes like *Dest*. (The usefulness of attributes in expressing elaboration tolerance was discussed in (Lifschitz 2000).)

In the following two sections, we compare $\mathcal{BC}$+ with $\mathcal{BC}$ and $\mathcal{C}$+.

Relation to Language $\mathcal{BC}$

Review: $\mathcal{BC}$

The signature σ 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 $\mathcal{BC}$ causal laws and $\mathcal{BC}$+ causal laws is that the former allows only the conjunction of atoms in the body, and distinguishes between **if** and **ifcons** clauses.

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

$$A_0 \textbf{ if } A_1, \dots, A_m \textbf{ ifcons } A_{m+1}, \dots, A_n \qquad (12)$$

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

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

$$A_0 \textbf{ after } A_1, \dots, A_m \textbf{ ifcons } A_{m+1}, \dots, A_n \qquad (13)$$

$(n \geq m \geq 0)$, where (i) A_0 is an atom containing a regular fluent constant, (ii) each of $A_1, \dots, A_m$ is an atom containing a fluent constant, or $a = \mathbf{t}$ where a is an action constant, and (iii) $A_{m+1}, \dots, A_n$ 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 $A_1, \dots, A_m$, and $A_{m+1}, \dots, A_n$ 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 by using 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 defined in Section "Semantics of $\mathcal{BC}$+."

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

- $$i{:}A_0 \ \leftarrow \ i : (A_1 \wedge \dots \wedge A_m \wedge \neg\neg A_{m+1} \wedge \dots \wedge \neg\neg A_n) \qquad (14)$$
  for every $\mathcal{BC}$ static law (12) in D and every $i \in \{0, \dots, m\}$;
- $$(i+1){:}A_0 \ \leftarrow \ i{:}(A_1 \wedge \dots \wedge A_m) \wedge (i{+}1){:}(\neg\neg A_{m+1} \wedge \dots \wedge \neg\neg A_n) \qquad (15)$$
  for every $\mathcal{BC}$ dynamic law (13) in D and every $i \in \{0, \dots, m{-}1\}$;
- the formula $i : (a = \mathbf{t} \vee a = \mathbf{f})$ for every action constant a and every $i \in \{0, \dots, m{-}1\}$;
- the formula (7) for every regular fluent constant c and every element $v \in \mathit{Dom}(c)$;
- the formulas (1) and (2) for every constant $i{:}c$ in $\sigma_{D,m}$.

Note how the translations (14) and (15) treat **if** and **ifcons** clauses differently by either prepending double negations in front of atoms or not. In $\mathcal{BC}$+, only one **if** clause is enough since the formulas are understood under the stable model semantics.

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 $\mathbf{bc2bcp}(D)$, which turns a $\mathcal{BC}$ description into an equivalent $\mathcal{BC}$+ description as follows:

- replace every causal law (12) with
  $$\textbf{caused } A_0 \textbf{ if } A_1 \wedge \dots \wedge A_m \wedge \neg\neg A_{m+1} \wedge \dots \wedge \neg\neg A_n; \qquad (16)$$
- replace every causal law (13) with
  $$\textbf{caused } A_0 \textbf{ if } \neg\neg A_{m+1} \wedge \dots \wedge \neg\neg A_n \textbf{ after } A_1 \wedge \dots \wedge A_m;$$
- add the causal laws "**exogenous** a" 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 the description $\mathbf{bc2bcp}(D)$ in language $\mathcal{BC}$+.*

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

In $\mathcal{BC}$, every action is assumed to be Boolean, and action dynamic laws are not available. This prevents us from describing defeasible causal laws (Giunchiglia et al. 2004, Section 4.3) and action attributes (Giunchiglia et al. 2004, Section 5.6), which $\mathcal{BC}$+ and $\mathcal{C}$+ are able to express naturally. Also, syntactically, $\mathcal{BC}$ is not expressive enough to describe dependencies 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 by the fluent dynamic law

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

but this 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 hide these constructs by using the **default** proposition in $\mathcal{BC}$+ as the following example illustrates.

Consider the leaking container example from (Lee, Lifschitz, and Yang 2013) in which a container loses k units of liquid by default. This example was used in that paper to illustrate the advantages of $\mathcal{BC}$ over $\mathcal{B}$ that is able to express defaults other than inertia. In this domain, the default decrease of *Amount* over time can be represented in $\mathcal{BC}$+ using the **default** proposition

$$\textbf{default } \mathit{Amount} = x \textbf{ after } \mathit{Amount} = x + k, \qquad (17)$$

which stands for fluent dynamic law

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

which can be further turned into propositional formulas

$$\{i{+}1 : \mathit{Amount} = x\}^{\mathrm{ch}} \leftarrow i : \mathit{Amount} = x + k \qquad (18)$$

$(i < m)$. On the other hand, the abbreviation (17) in $\mathcal{BC}$ stands for the causal law

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

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

Regular fluent constants:	Domains:
$Status(s)$	$\{Up, Dn\}$
Action constants:	**Domains:**
$Flip(s)$	Boolean

Causal laws:

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

Figure 2: Two Switches

which is further turned into

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

($i < m$), which is strongly equivalent to (18).

Our experience is that, in most cases, instead of remembering the subtle difference between **if** and **if cons** clauses in $\mathcal{BC}$, it is more intuitive to use the **default** proposition in $\mathcal{BC}$+.

Relation to $\mathcal{C}$+

Due to the space limitation, we refer the reader to (Giunchiglia et al. 2004) for the review of $\mathcal{C}$+. The syntax of $\mathcal{C}$+ has the same form of causal laws as $\mathcal{BC}$+.

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

We say that $\mathcal{C}$+ action description D is *definite* if the head of every causal law is either $\bot$ or an atom $c=v$. For any definite $\mathcal{C}$+ description D, we define the translation $\mathbf{cp2bcp}(D)$, which turns a $\mathcal{C}$+ description into $\mathcal{BC}$+, as follows:

- replace every $\mathcal{C}$+ causal law (3) with

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

- replace every $\mathcal{C}$+ causal law (4) 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 the description* $\mathbf{cp2bcp}(D)$ *in language $\mathcal{BC}$+.*

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

Recall that the syntax of $\mathcal{BC}$+ is identical to 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 usable in the context of action languages. We already observed that being able to use aggregates in $\mathcal{BC}$+ provides a succinct representation of the Blocks World domain. However, the expansion of choice formulas and aggregates into propositional formulas as in the stable model semantics does not work in $\mathcal{C}$+ as defined in (Giunchiglia et al. 2004), which is based on nonmonotonic causal theories.

The embedding of $\mathcal{C}$+ in $\mathcal{BC}$+ tells us that **if** clauses always introduce double negations, whose presence do not necessarily lead to minimal models. This accounts for the fact that the definite fragment of $\mathcal{C}$+ does not handle the concept of transitive closure correctly. The inability to consider minimal models in such cases introduce some unintuitive behavior of $\mathcal{C}$+ in representing causal dependencies among 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 Dn and $Switch_2$ is Up:

$\langle \{St(Sw_1)=Dn, St(Sw_2)=Up\},$
$\quad \{\sim Flip(Sw_1), \sim Flip(Sw_2)\}, \{St(Sw_1)=Dn, St(Sw_2)=Up\}\rangle,$
$\langle \{St(Sw_1)=Dn, St(Sw_2)=Up\},$
$\quad \{Flip(Sw_1), \sim Flip(Sw_2)\}, \{St(Sw_1)=Up, St(Sw_2)=Dn\}\rangle,$
$\langle \{St(Sw_1)=Dn, St(Sw_2)=Up\},$
$\quad \{\sim Flip(Sw_1), Flip(Sw_2)\}, \{St(Sw_1)=Up, St(Sw_2)=Dn\}\rangle,$
$\langle \{St(Sw_1)=Dn, St(Sw_2)=Up\},$
$\quad \{Flip(Sw_1), Flip(Sw_2)\}, \{St(Sw_1)=Up, St(Sw_2)=Dn\}\rangle.$

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,

$\langle \{St(Sw_1)=Dn, St(Sw_2)=Up\},$
$\quad \{\sim Flip(Sw_1), \sim Flip(Sw_2)\}, \{St(Sw_1)=Up, St(Sw_2)=Dn\}\rangle$

is also a transition because, according to the semantics of $\mathcal{C}$+, this is causally explained by the cyclic causality. Obviously, this is unintuitive.

Conclusion

We implemented language $\mathcal{BC}$+ by modifying system 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 similar to the translation $PF_m(D)$ for $\mathcal{BC}$+, modifying this translation to accept $\mathcal{BC}$+ input is straightforward.

Besides the action languages considered in this paper, language $\mathcal{K}$ (Eiter et al. 2004) is also defined via ASP. Since $\mathcal{K}$ assumes knowledge-states, it is not directly comparable to $\mathcal{BC}$+ which assumes world-states.

A further extension of $\mathcal{BC}$+ is possible. It is straightforward to extend $\mathcal{BC}$+ to the first-order level, by using the first-order stable model semantics from (Ferraris, Lee, and Lifschitz 2011) or its extension with generalized quantifiers (Lee and Meng 2012) in place of propositional formulas. This will allow $\mathcal{BC}$+ to include more general constructs, such as constraint atoms, external atoms, nonmonotonic dl-atoms, as they are instances of generalized quantifiers as shown in (Lee and Meng 2012).

Acknowledgements We are grateful to Michael Bartholomew, Vladimir Lifschitz, and the anonymous referees for their useful comments. This work was partially supported by the National Science Foundation under Grant IIS-1319794 and by the South Korea IT R&D program MKE/KIAT 2010-TD-300404-001, and the Brain Pool Korea program.

References

Babb, J., and Lee, J. 2013. Cplus2ASP: Computing action language $\mathcal{C}+$ in answer set programming. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 122–134.

Bartholomew, M., and Lee, J. 2014. Stable models of multi-valued formulas: partial vs. total functions. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 583–586.

Eiter, T.; Faber, W.; Leone, N.; Pfeifer, G.; and Polleres, A. 2004. A logic programming approach to knowledge-state planning: Semantics and complexity. *ACM Trans. Comput. Log.* 206–263.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2011. Stable models and circumscription. *Artificial Intelligence* 175:236–263.

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

Gelfond, M., and Lifschitz, V. 1993. Representing action and change by logic programs. *Journal of Logic Programming* 17:301–322.

Gelfond, M., and Lifschitz, V. 1998. Action languages.[5] *Electronic Transactions on Artificial Intelligence* 3:195–210.

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

Giunchiglia, E.; Lee, J.; Lifschitz, V.; McCain, N.; and Turner, H. 2004. Nonmonotonic causal theories. *Artificial Intelligence* 153(1–2):49–104.

Harrison, A. J.; Lifschitz, V.; and Yang, F. 2014. The semantics of gringo and infinitary propositional formulas. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR*.

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

Lee, J., and Meng, Y. 2012. Stable models of formulas with generalized quantifiers.[6] In *Proceedings of International Workshop on Nonmonotonic Reasoning (NMR)*.

Lee, J.; Lifschitz, V.; and Yang, F. 2013. Action language $\mathcal{BC}$: Preliminary report. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

Lifschitz, V. 2000. Missionaries and cannibals in the Causal Calculator. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 85–96.

Pelov, N.; Denecker, M.; and Bruynooghe, M. 2003. Translation of aggregate programs to normal logic programs. In *Proceedings Answer Set Programming*.

Son, T. C., and Pontelli, E. 2007. A constructive semantic characterization of aggregates in answer set programming. *TPLP* 7(3):355–375.

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

[6] `http://peace.eas.asu.edu/joolee/papers/smgq-nmr.pdf`