CF2-extensions as Answer-set Models

Juan Carlos Nieves,**Ignasi Gómez-Sebastià**
Universitat Politècnica de Catalunya (UPC)
{jcnieves,igomez}@lsi.upc.edu
and
Mauricio Osorio
Universidad de las Américas - Puebla
osoriomauri@googlemail.com

COMMA'10, September 8, 2010, Desenzano del Garda, Italy

Outline

Extension-based argumentation semantics

- Extension-based argumentation semantics formalized in terms of relationships between atomic arguments. Characterizable in terms of answer-sets.
- This work presents an approach for characterizing SCC-recursive semantics (CF2) in terms of answer set models.
- Not only extension-based argumentation semantics based on admissible sets can be characterized in terms of answer sets, but also extension-based argumentation semantics based on SCC.

Abstract argumentation semantics I

Dung[1]. presented an unifying framework which has played an influential role on argumentation research and Artificial Intelligence.
Abstract argumentation semantics: *Grounded, Stable, Preferred and Complete* semantics. Based on the concept of admissible set: *Coherent point of view from a conflicting set of arguments*
Formally:

An argumentation framework is a pair $AF = \langle AR, attacks \rangle$, where AR is a set of arguments, and *attacks* is a binary relation on AR, i.e. $attacks \subseteq AR \times AR$. A attacks B (or B is attacked by A) if $attacks(A, B)$ holds.

[1] P. M. Dung. **On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games.** *Artificial Intelligence*, 77(2):321-358, 1995.

Abstract argumentation semantics II

- A set S of arguments is conflict-free if there are no arguments A, B in S such that A *attacks* B.
- An argument $A \in AR$ is acceptable with respect to a set S of arguments if and only if for each argument $B \in AR$: If B attacks A then B is attacked by S.
- A conflict-free set of arguments S is admissible if and only if each argument in S is acceptable *w.r.t.* S.

Abstract augmentation semantics and Logic programming semantics

Dung proved an interesting result *w.r.t.* the grounded and stable semantics. In particular, he proved that:

- The grounded semantics can be characterized by the well-founded semantics and
- The stable semantics can be characterized by the answer set semantics.

This result points out the existence of a direct relationship between argumentation semantics and logic programming semantics.
As consequences of this result, we have that:

Observations

We can study abstract augmentation semantics based on logic programming semantics. This means that a proper logic programming semantics can describe the interaction of conflicting arguments.

SCC-recursive semantics I

Approach based on splitting an argumentation framework into several strongly-connected components, which are processed separately:

Sets of arguments whose status does not depend on the status of arguments outside the set

An order (directionality principle) is defined between the different sets of strongly-connected components.

Given two strongly connected components SCC_A and SCC_B it is said that SCC_A precedes SCC_B if there is a directed path in $\langle AR, attacks \rangle$ from any argument in SCC_A to an argument in SCC_B.

Strongly-connected components are processed on a recursive way, thus, the results of processing a component affects the following ones.

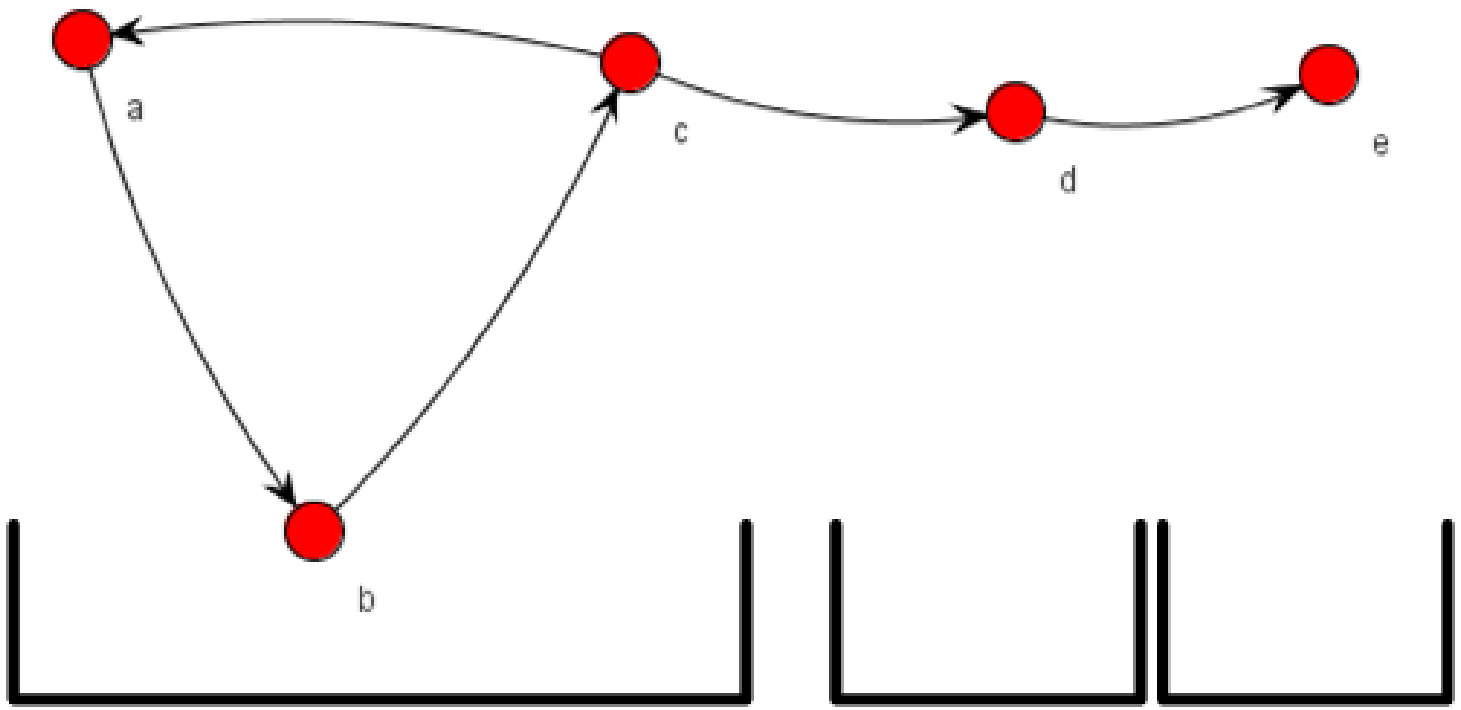
SCC-recursive semantics II

The approach is:

- Powerful: able to process semantics on argumentation frameworks where other approaches do not show proper behaviors(preferred on frameworks with odd-length cycles)
- Versatile: the base function of the recursive approach is left open. One can define a wide range of functions to capture a wide range of different semantics.

SCC-recursive semantics III

SCC Example:



SCC1 < SCC2 < SCC3

General algorithm

1. Inferring the grounded extension
2. Reducing the given argumentation framework by considering the inferred grounded extension
   - In this case, reduction is performed based on a labeling approach, labeling each argument as: accepted, defeated or undefined
3. If the reduced argumentation framework has undefined arguments, the base function is applied and the process follows in Step 1

SCC-recursive semantics via Answer sets I

Characterizing the argumentation framework as a logic program. *N* in *int(N)* manages an iteration for inferring the grounded extension. *T* in *time(T)* manages an iteration in the overall process of inferring the SCC-recursive semantics.

$$
\begin{array}{ll}
\Pi_{arg} = & \{arg(a,0) \leftarrow \top | a \in AR\}. \\
\Pi_{at} = & \{at(a,b) \leftarrow \top | (a,b) \in attacks\}. \\
\Pi_{int} = & \{\ time(0) \leftarrow \top; \ldots; time(n+1) \leftarrow \top; \\
 & int(0) \leftarrow \top; \ldots; int(n) \leftarrow \top; size(n) \leftarrow \top | n \text{ is the cardinality of } AR\}
\end{array}
$$

```
ar(d).
ar(e).
ar(b).
ar(c).
ar(a).
at(d,e).
at(b,c).
at(c,d).
at(c,a).
at(a,b).
```

SCC-recursive semantics via Answer sets II

Characterization of the grounded semantics in terms of answer-set models:

$$
\begin{aligned}
\Pi_{GE} = \quad & a\_gr(X,0,T1) \leftarrow arg(X,T), T1 = T+1,\ not\ not\_a\_gr(X,T).\\
& not\_a\_gr(X,T) \leftarrow arg(X,T), at(Y,X),\ not\ d(Y,T), time(T).\\
& a\_gr(X,N,T) \leftarrow int(N), N > 0, arg(X,T),\ not\ not\_a\_gr\_d(X,N,T)\\
& \qquad\qquad\qquad\qquad\qquad\qquad , time(T).\\
& not\_a\_gr\_d(X,N,T) \leftarrow at(Y,X), N = M+1,\ not\ attacked(Y,M,T),\\
& \qquad\qquad\qquad\qquad\qquad\qquad arg(X,T), arg(Y,T), time(T).\\
& attacked(Y,M,T) \leftarrow arg(Y,T), arg(Z,T), at(Z,Y), a\_gr(Z,M,T), time(T).
\end{aligned}
$$

SCC-recursive semantics via Answer sets III

Defining the states of an argument:

- ac_gr: Accepted, belongs to the grounded extension.
- d_gr: Defeated, attacked by an argument that belongs to the grounded extension.
- i_gr: undefined, neither accepted nor defeated.

$$
\begin{aligned}
\Pi_{gr\_states} = \quad & ac\_gr(X,T) \leftarrow a\_gr(X, fixp, T), time(T), size(fixp).\\
& d\_gr(X,T) \leftarrow at(Y,X), ac\_gr(Y,T), arg(Y,T), arg(X,T), time(T).\\
& i\_gr(X,T) \leftarrow arg(X,T),\ not\ ac\_gr(X,T),\ not\ d\_gr(X,T), time(T).\\
& at\_d(X,Y,T) \leftarrow arg(X,T), arg(Y,T), at(X,Y),\\
& \qquad not\ not\_at\_d(X,Y,T), time(T).\\
& not\_at\_d(X,Y,T) \leftarrow arg(X,T), arg(Y,T), at(X,Y), d\_gr(X,T)\\
& \qquad , time(T).
\end{aligned}
$$

SCC-recursive semantics via Answer sets IV

Defining the restricted argumentation framework. Identifying the existing *strongly-connected components* and applying directionality principle.

$$
\begin{aligned}
\Pi_{AF\downarrow_{i\_gr}} = \quad & base(X,T) \vee other\_base(X,T) \leftarrow i\_gr(X,T), time(T).\\
& \leftarrow base(X,T), base(Y,T), X != Y, not\ cycle(X,Y,T), time(T).\\
& \leftarrow base(X,T), ar(Y), X != Y, other\_base(Y,T), cycle(X,Y,T).\\
& \leftarrow not\_base(T), time(T).\\
& \leftarrow incomplete(T), time(T).\\
& incomplete(T) \leftarrow i\_gr(X,T), not\ not\_empty(T), time(T).\\
& not\_empty(T) \leftarrow base(X,T), time(T).\\
& not\_base(T) \leftarrow base(X,T), arg(Y,T), other\_base(Y,T), at(Y,X)\\
& \qquad\qquad , time(T).\\
& cycle(X,Y,T) \leftarrow path(X,Y,T), path(Y,X,T), time(T).\\
& path(X,Y,T) \leftarrow at(X,Y), arg(X,T), arg(Y,T), time(T).\\
& path(X,Y,T) \leftarrow arg(X,T), arg(Y,T), arg(Z,T), at(X,Z),\\
& \qquad\qquad path(Z,Y,T), time(T).
\end{aligned}
$$

SCC-recursive semantics via Answer sets V

The status of the arguments (mainly accepted or defeated) have to be preserved over the different restricted argumentation frameworks

$$
\begin{aligned}
\Pi_{inertial} = \quad & ac(X, T1) \leftarrow T1 = T + 1, ac(X, T), time(T1). \\
& d(X, T1) \leftarrow T1 = T + 1, d(X, T), time(T1). \\
& ac(X, T) \leftarrow ac\_gr(X, T), time(T). \\
& d(X, T) \leftarrow d\_gr(X, T), time(T). \\
& arg(X, T1) \leftarrow arg(X), T1 = T + 1, not\ ac(X, T), notd(X, T), time(T) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad , time(T1).
\end{aligned}
$$

SCC-recursive semantics via Answer sets VI

Define a base function which infers maximal conflict-free sets of arguments taking into account the directionality principle. This allows to characterize CF2 argumentation semantics.

$$
\begin{aligned}
\Pi_{base\_function} = \quad & d(X,T) \leftarrow ac(Y,T), at(Y,X), time(T).\\
& ac(X,T) \leftarrow base(X,T), not\ d(X,T), time(T).\\
& d(X,T) \leftarrow at(Y,X), base(X,T), base(Y,T), ac(Y,T), time(T).\\
& d(X,T) \vee d(Y,T) \leftarrow at(Y,X), path\_d(X,Y,T), base(X,T),\\
& \qquad\qquad base(Y,T), time(T).\\
& path\_d(X,Y,T) \leftarrow at(X,Y), base(X,T), base(Y,T), time(T).\\
& path\_d(X,Y,T) \leftarrow at(X,Z), base(X,T), base(Y,T), base(Z,T),\\
& \qquad\qquad path\_d(Z,Y,T), time(T).\\
& accepted(X) \leftarrow ac(X,fixp), size(fixp).\\
& a\_gr(X,0,T1) \leftarrow T1 = T+1, ac(X,T)
\end{aligned}
$$

SCC-recursive semantics via Answer sets VII

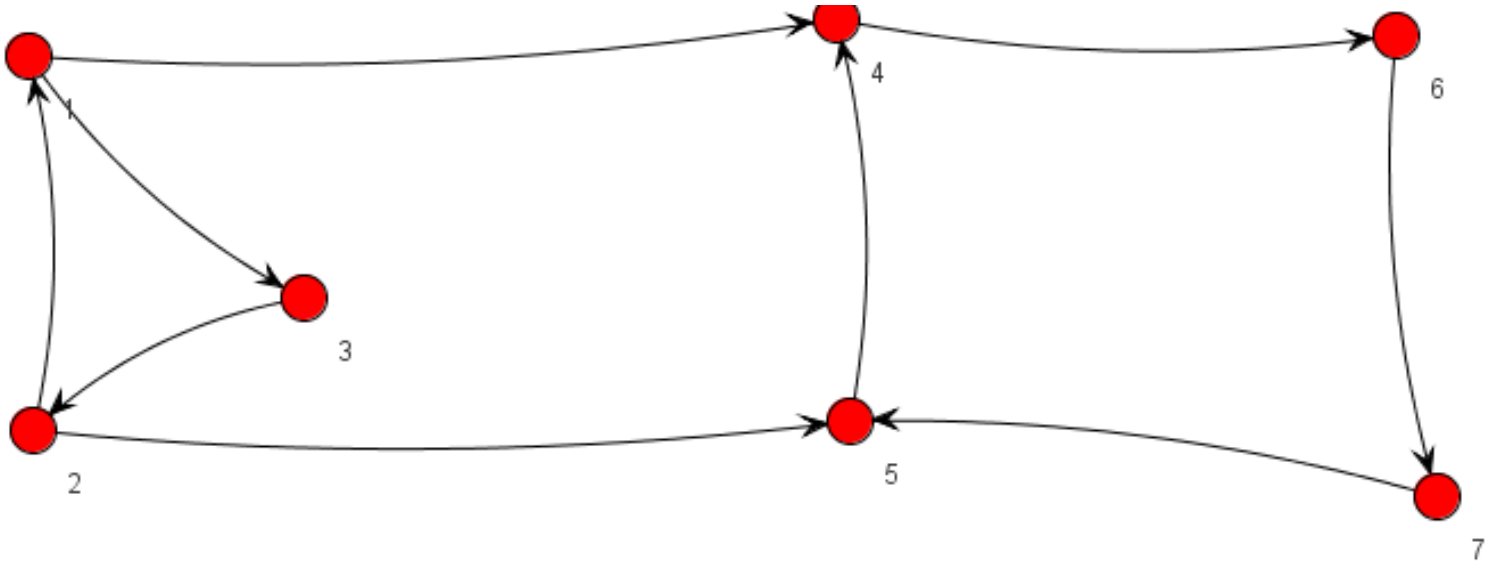
We can take advantadge of efficient answer-set solvers for answering the following questions w.r.t CF2 semantics on any Argumentation Framework *AF*:

- Which are the extensions in S(AF)?
- Is an argument A accepted in at least one extension in S(AF)?
- Is an argument A accepted in every extension in S(AF)?
- Which are the extensions in S(AF) where a given argument is accepted?
- Which are the extensions in S(AF) where a given argument is defeated?

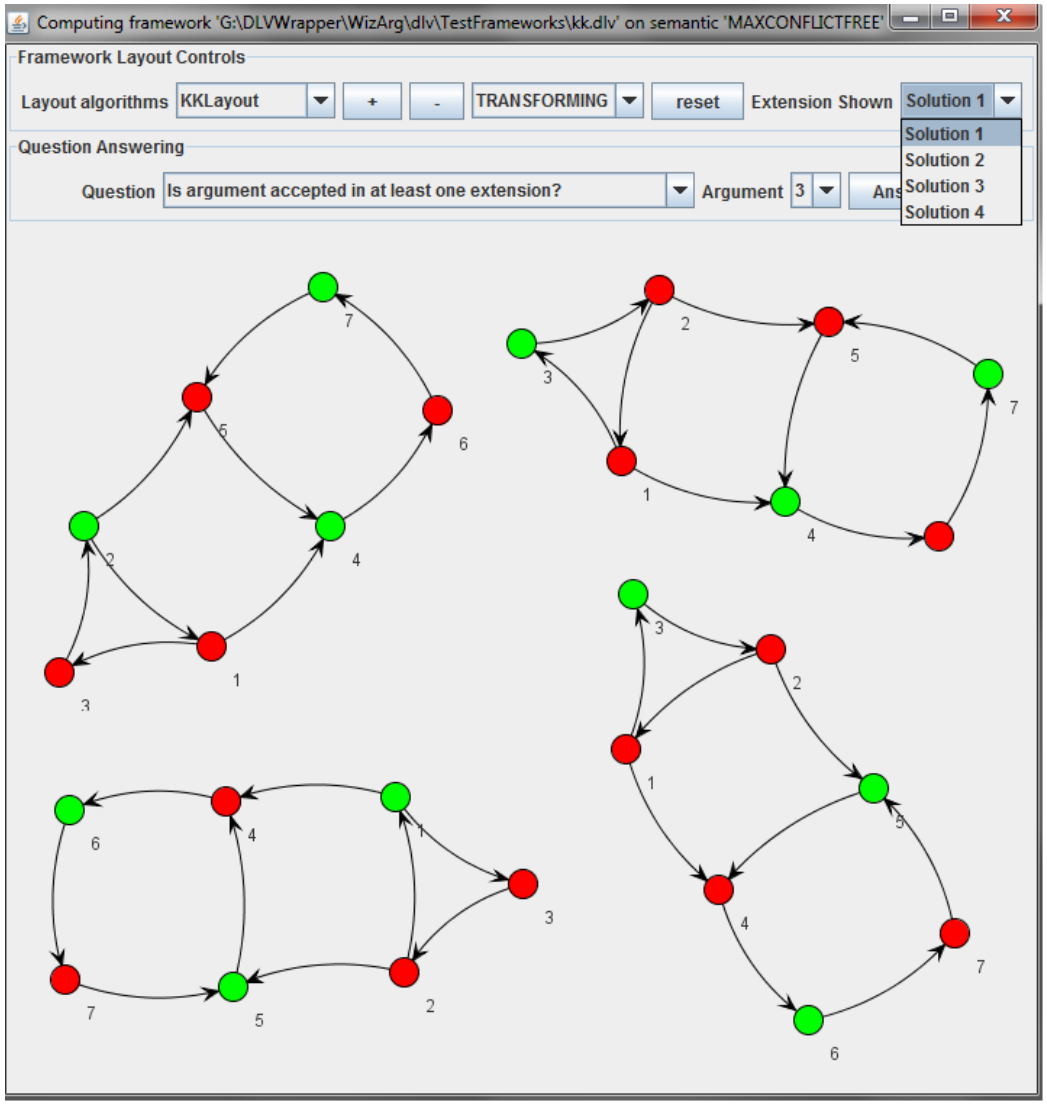
SCC-recursive semantics via Answer sets VII

- Able to represent graphically and interactively argumentation frameworks
  - Interactively design argumentation frameworks as directed graphs
  - Visualize the different solutions of applying a framework to a semantics
  - Answer the questions mentioned before
- Able to integrate in Java processes logic programs and meta-interpreters
  - The Java process is able to solve frameworks under different semantics (CF2, Grounded, etc.)

Example: Given an argumentation framework

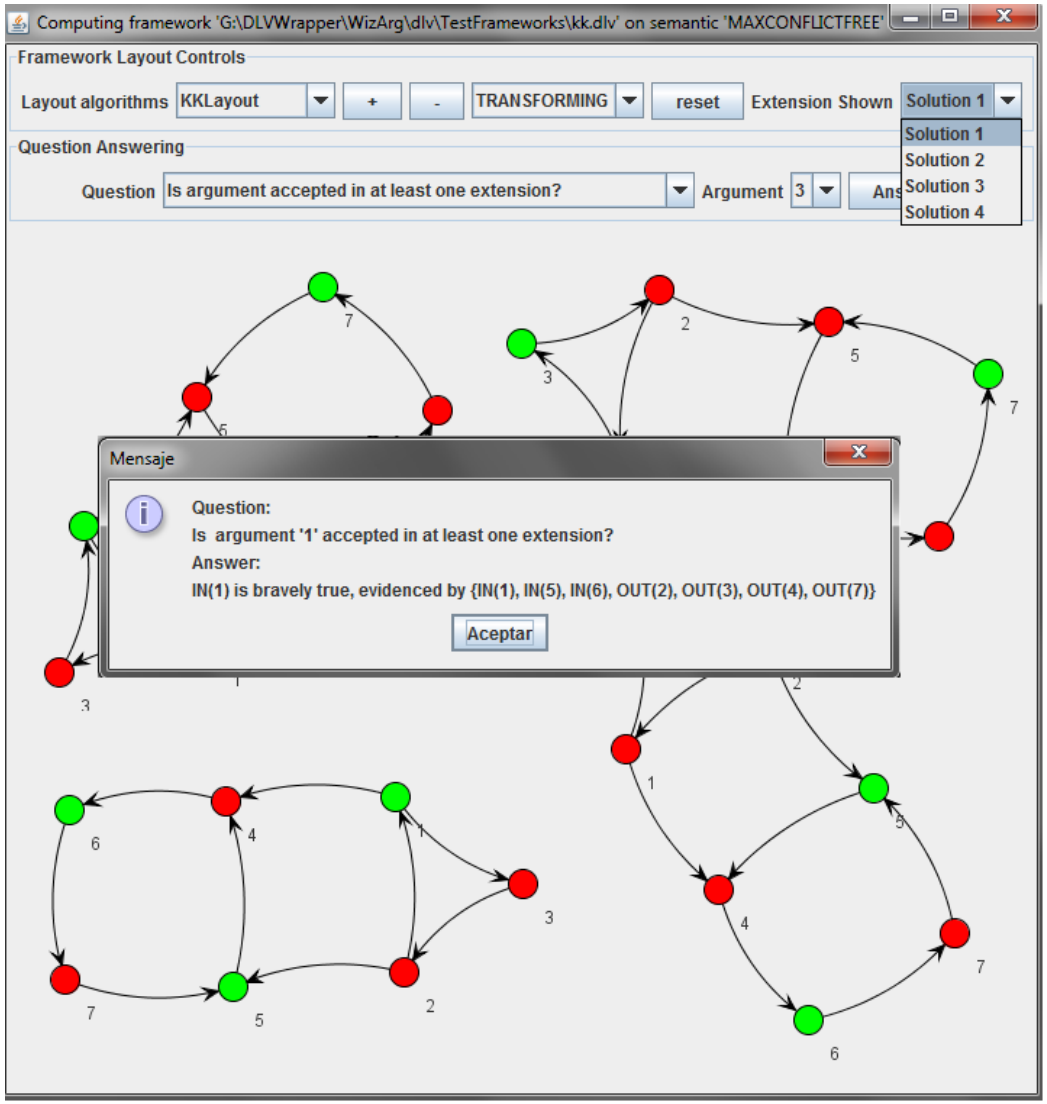


Example: Which are the extensions in S(AF)?

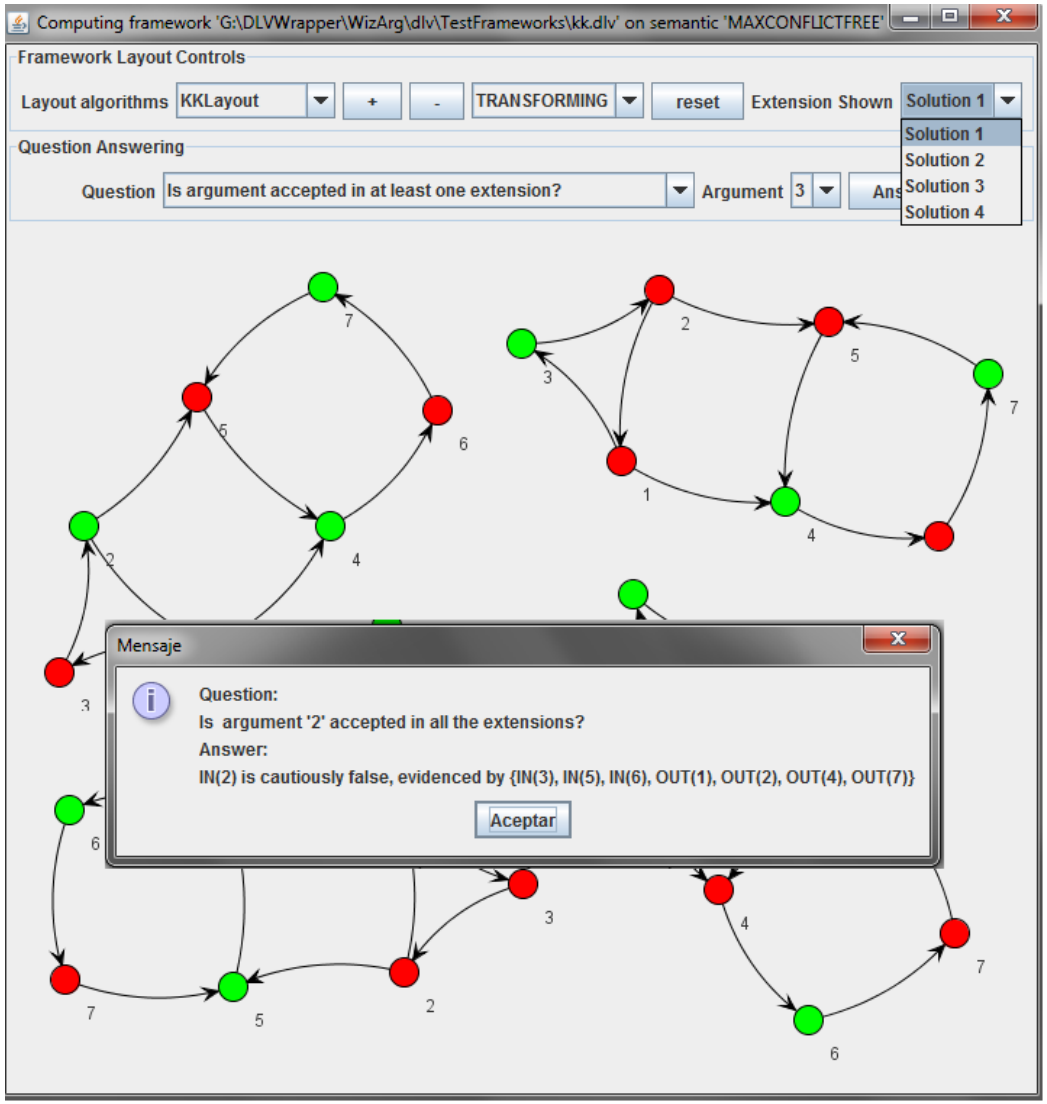


An example

Example: Is an argument A accepted in at least one extension in S(AF)?

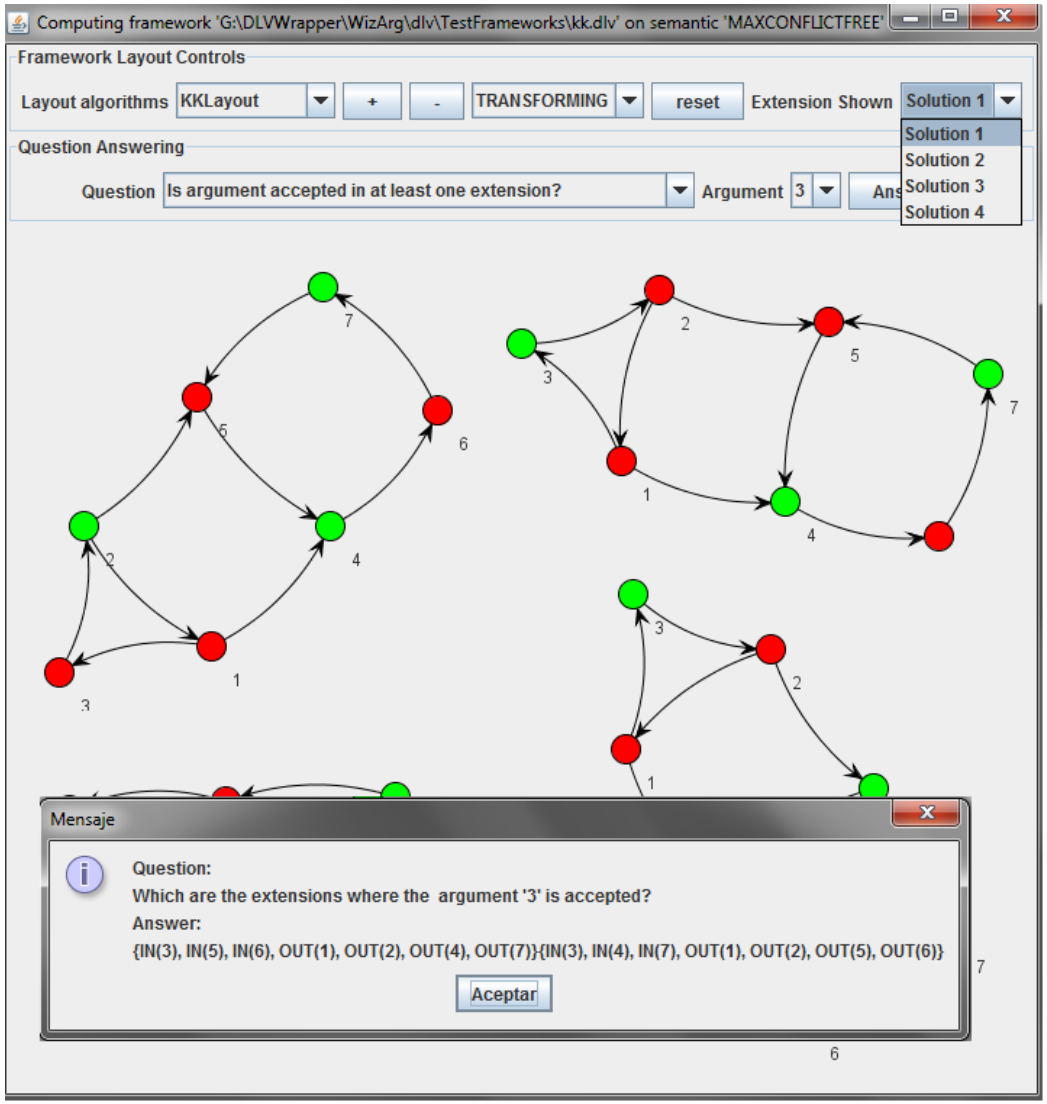


Example: Is an argument A accepted in every extension in S(AF)?



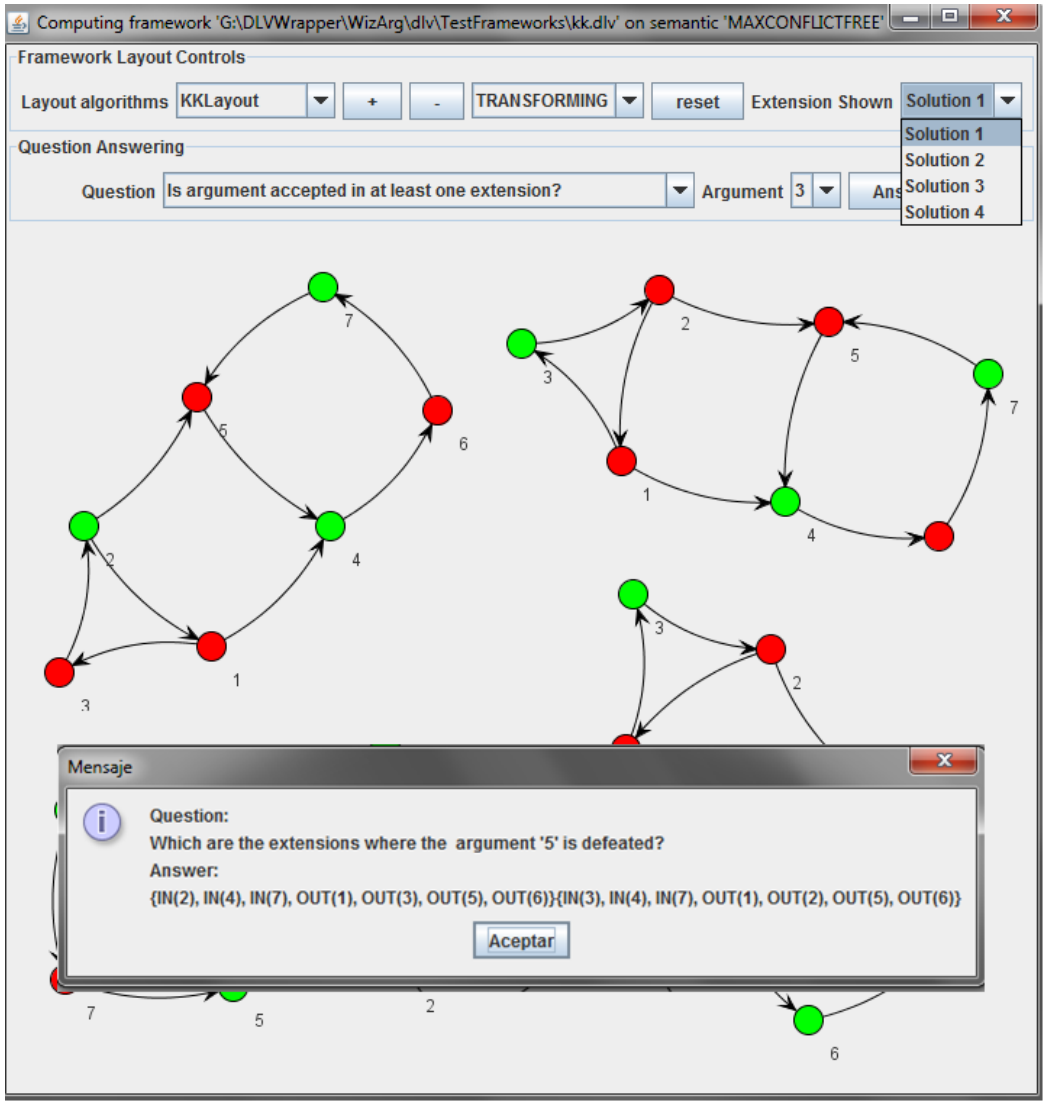
An example

Which are the extensions in S(AF) where a given argument is accepted?



An example

Which are the extensions in S(AF) where a given argument is defeated?



Conclusions

- Extension-based argumentation semantics based on admissible sets are able to be characterized in terms of answer set models. However, it is well-known they have unexpected behaviors
- Need to evaluate emerging approaches. Several approaches available. Ones based on SCC-recursive semantics look sound.
- CF2 semantics converge with extension-based argumentation semantics based on logic programming semantics with negation as failure
- We have shown not only extension-based argumentation semantics based on admissible sets can be characterized in terms of answer set models; but also extension-based argumentations semantics based on strongly connected components can be characterized in terms of answer set models

Conclusions

Building an interpreter that processes the argumentation framework as an input (in contrast to using a fixed logic program that depends on the argumentation framework) provides the following benefits:

- The interpreter is easier to understand, extend and debug.
- Eases the process of providing the correspondence between answer sets and extensions.
- The interpreter is independent from the argumentation framework. Once can change the framework (or even create a framework library) without making changes on the interpreter.

Farewell

Thank you for your attention!
Resources available on the net:

- {jcnieves,igomez}@lsi.upc.edu and osoriomauri@googlemail.com
- http://www.lsi.upc.edu/ jcnieves/software/CF2-ASP.dlv
- http://sourceforge.net/projects/wizarg/