SESAME - A System for Specifying Semantics in Abstract Argumentation

Philippe Besnard, Sylvie Doutre, van Hieu Ho, Dominique Longin

SESAME – A System for Specifying Semantics in Abstract Argumentation

Philippe BESNARD [a], Sylvie DOUTRE [b,1], Van Hieu HO [c] and Dominique LONGIN [a]
[a] *IRIT, CNRS*
[b] *IRIT, University of Toulouse 1*
[c] *IRIT, University of Toulouse 3*

Abstract. This is a report on an implemented system that allows the user to specify an argumentation semantics and that returns a logical encoding in the form of a parameterized propositional formula. When applied to a subset S of a given argumentation graph G, the instantiated formula is satisfiable if and only if S is an extension for G according to the argumentation semantics specified by the user.

Keywords. Abstract argumentation, argumentation semantics, logical encoding.

1. Introduction

This paper is a report on SESAME (SEmantics Specification for Abstract arguMEntation), a system that allows the user to specify an argumentation semantics and that returns a logical encoding in the form of a parameterized propositional formula [1]. When applied to a given argumentation graph $G = (\mathscr{A}, \mathfrak{R})$, for $S \subseteq \mathscr{A}$, the instantiated formula is satisfiable if and only if S is an extension for G according to the argumentation semantics specified by the user.

The principle of such an encoding was introduced in [1] (see also [2,3], see [4] for an ASP variant) where a propositional formula $\sigma_{(\mathscr{A},\mathfrak{R}),S}$ is defined that turns out to be satisfiable if and only if S is a σ-extension of the argumentation graph $(\mathscr{A}, \mathfrak{R})$. That is, the formula $\sigma_{(\mathscr{A},\mathfrak{R}),S}$ encodes the argumentation semantics σ.

Let us consider e.g. the stable semantics [5] and construct $\sigma_{(\mathscr{A},\mathfrak{R}),S}$ for $\sigma = \mathit{stable}$.

Definition 1. *Given an argumentation graph $G = (\mathscr{A}, \mathfrak{R})$, a* stable extension *$S \subseteq \mathscr{A}$ is a set that satisfies the following two conditions:*

1. *there does not exist two arguments a and b in S such that $a\mathfrak{R}b$;*
2. *for each argument $a \notin S$, there exists $b \in S$ such that $b\mathfrak{R}a$.*

That a set of arguments S satisfies the first condition amounts to: *for all a in S, for all b attacking a, it is not the case that b is in S*. In symbols,

$$\forall a \in S \; \forall b \mathfrak{R} a \quad b \notin S$$

[1]Corresponding Author: Sylvie Doutre, University of Toulouse 1, Toulouse, France; E-mail: sylvie.doutre@irit.fr.

The statement $b \in S$ can be encoded[2] through identifying b with a propositional symbol (letting the propositional symbol b to be read "b is in S"), hence an encoding for the statement that S satisfying condition 1. is

$$\bigwedge_{a \in S} \bigwedge_{b \mathfrak{R} a} \neg b \tag{1}$$

As to the second condition for S to be a stable extension, it can be expressed by: *for all a in $\mathscr{A}$, if a is not in S then there exists some b attacking a such that b is in S.*

$$\forall a \in \mathscr{A} \left(a \notin S \;\Rightarrow \left(\exists b \mathfrak{R} a \;\; b \in S \right) \right)$$

The statement $b \in S$ is again to be encoded via identifying b with a propositional symbol, and, similarly, $a \in S$ is encoded as a. Hence, an encoding for the statement that S satisfies condition 2. is

$$\bigwedge_{a \in \mathscr{A}} \left(\neg a \rightarrow \bigvee_{b \mathfrak{R} a} b \right) \tag{2}$$

Conjoining (1) and (2), we obtain a subformula (of $\sigma_{(\mathscr{A},\mathfrak{R}),S}$) expressing that S meets conditions 1. and 2. for being a stable extension:

$$\left(\bigwedge_{a \in S} \bigwedge_{b \mathfrak{R} a} \neg b \right) \;\; \wedge \;\; \left(\bigwedge_{a \in \mathscr{A}} \left(\neg a \rightarrow \bigvee_{b \mathfrak{R} a} b \right) \right) \tag{3}$$

There remains to introduce the set Φ_S of literals expressing that all elements of S are in S and that all elements not in S fail to be in S, as follows

$$\Phi_S \stackrel{\text{def}}{=} \{a \mid a \in S\} \cup \{\neg a \mid a \in \mathscr{A} \setminus S\}$$

and all this leads us to a formula encoding stable extensions:

$$\sigma_{(\mathscr{A},\mathfrak{R}),S} = \left(\bigwedge_{a \in S} \bigwedge_{b \mathfrak{R} a} \neg b \right) \;\; \wedge \;\; \left(\bigwedge_{a \in \mathscr{A}} \left(\neg a \rightarrow \bigvee_{b \mathfrak{R} a} b \right) \right) \;\; \wedge \;\; \bigwedge \Phi_S \tag{4}$$

Thus, the system requires the user to input an argumentation semantics as a combination of principles (see [6] for the notion) —excluding $\bigwedge \Phi_S$ which is transparent to the user.

Notation. *Quantifying over attacks relative to an argument can easily be turned into quantifying over membership in $\mathfrak{R}^+(\cdot)$ and $\mathfrak{R}^-(\cdot)$ according to the next definition:*

$\mathfrak{R}^+(a) \stackrel{\text{def}}{=} \{b \in \mathscr{A} \mid a \mathfrak{R} b\}$;
$\mathfrak{R}^-(a) \stackrel{\text{def}}{=} \{b \in \mathscr{A} \mid b \mathfrak{R} a\}$.

Actually, the formula encoding stable extensions can be rewritten in many ways, e.g.

$$\sigma_{(\mathscr{A},\mathfrak{R}),S} = \bigwedge_{a \in S} \left(a \wedge \bigwedge_{b \in \mathfrak{R}^-(a)} \neg b \right) \wedge \bigwedge_{a \in \mathscr{A}} \left(\neg a \rightarrow \bigvee_{b \in \mathfrak{R}^-(a)} b \right) \wedge \bigwedge \{\neg a \mid a \in \mathscr{A} \setminus S\} \tag{5}$$

[2]Logical encoding of a statement "the argument b is in the set X" is denoted $\varphi_{(b \in X)}$ in [3] for genericity.

2. Logical Encoding of Argumentation Semantics

For a given semantics, $\sigma_{(\mathscr{A},\mathfrak{R}),S}$ represents a class of formulas each of which is induced from $\sigma_{(\mathscr{A},\mathfrak{R}),S}$ by considering a particular argumentation graph $(\mathscr{A},\mathfrak{R})$ and $S \subseteq \mathscr{A}$. It just takes to unfold $\sigma_{(\mathscr{A},\mathfrak{R}),S}$ according to the actual values taken by $\mathscr{A}$, $\mathfrak{R}$ and S.

Example 1. *As to σ=stable, for $\mathscr{A} = \{d,e,f\}$ with $\{e\mathfrak{R}f, f\mathfrak{R}d, f\mathfrak{R}f\}$ (see Figure 1), for the case $S = \{d,e\}$ which is a stable extension, instantiating $\sigma_{(\mathscr{A},\mathfrak{R}),S}$ (using the version of Equation (5) for readability reasons) yields the satisfiable formula:*

$$\overbrace{\Big((d \wedge \neg f) \wedge (e \wedge \top)\Big)}^{\bigwedge_{a\in S}\left(a \wedge \bigwedge_{b\in\mathfrak{R}^-(a)} \neg b\right)} \wedge \overbrace{\Big(\big(\neg d \rightarrow f\big) \wedge \big(\neg e \rightarrow \bot\big) \wedge \big(\neg f \rightarrow (e \vee f)\big)\Big)}^{\bigwedge_{a\in\mathscr{A}}\left(\neg a \rightarrow \bigvee_{b\in\mathfrak{R}^-(a)} b\right)} \wedge \overbrace{\Big(\neg f\Big)}^{\bigwedge\{\neg a \mid a \in \mathscr{A}\setminus S\}}$$

where $\top$ (resp. $\bot$) results from the empty conjunction $\bigwedge_{b\in\mathfrak{R}^-(e)} \neg b$ (resp. the empty disjunction $\bigvee_{b\in\mathfrak{R}^-(e)} b$) because no argument in $\mathscr{A}$ attacks e.

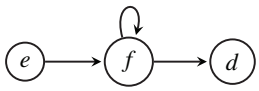


Figure 1. The argumentation graph $(\mathscr{A},\mathfrak{R})$ of Example 1

In $\sigma_{(\mathscr{A},\mathfrak{R}),S}$, *all* the atomic formulas encode a statement of the form "the argument a is in the set X" where a is a variable denoting an argument of $\mathscr{A}$ and X is a variable denoting a subset of $\mathscr{A}$. These are the *only* primitive statements that can appear in the semantics specified by the user. In the semantics σ, such statements can be conjuncted, negated, *etc*, arguments can be quantified over, sets of arguments can be maximized, *etc*.

Most importantly, S denotes a distinguished set: S denotes *the* subset of $\mathscr{A}$ to be tested for being a σ-extension.

3. Introducing the System

The main step in specifying an argumentation semantics via SESAME is developing a decomposition tree (from the grammar given in Section 3.1). This can be done by the user, by clicking upon the buttons in the left part of the screen (they are called *development buttons* – see Figure 2). When the user clicks upon one of these buttons, the current non-terminal[3] is expanded according to the derivation labelling the button.

As an illustration, when the user clicks upon the button ⟨principle⟩, a scroll menu is displayed that indicates what chains are available to replace ⟨principle⟩, namely "it isn't the case that ⟨principle⟩" or "⟨principle⟩ and ⟨principle⟩" or ... (see Figure 3).

After the user has selected her choice, the system updates the current expression (in semi-natural language) of the decomposition tree displayed in the main white frame. The formula[4] to be generated (it is displayed in the lower white frame) is duly updated.

[3]The non-terminals are the phrases enclosed between ⟨ and ⟩ (see the grammar).

[4]It is rather a pseudo-formula containing items such as ⟨principle⟩ as long as the user is not finished with specifying the argumentation semantics of hers.

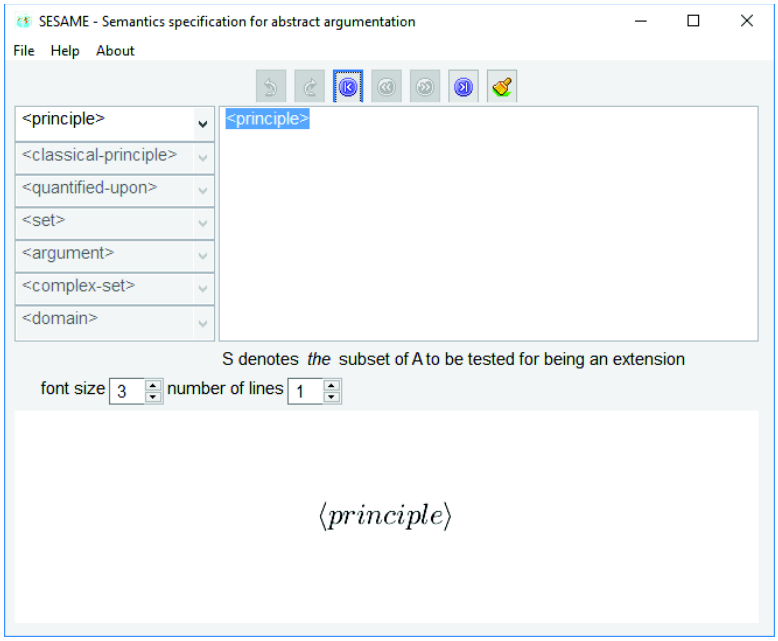


Figure 2. The home screen

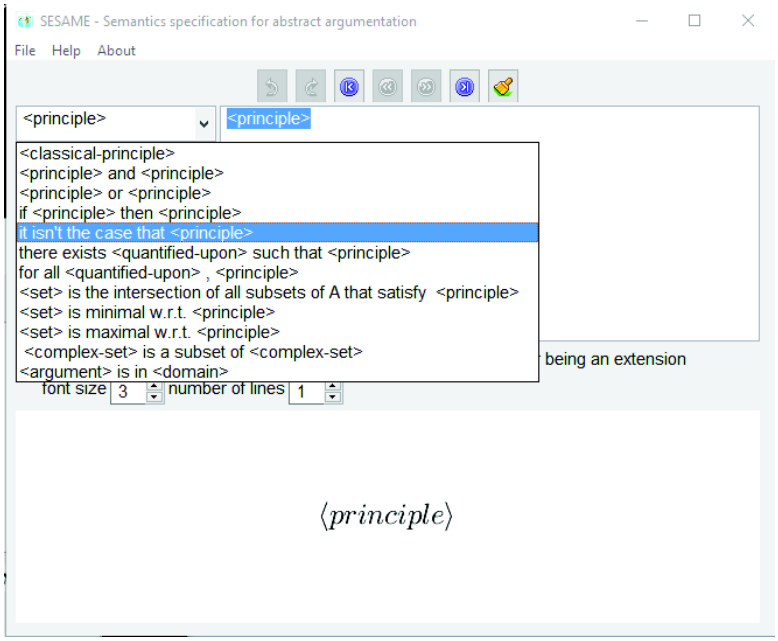


Figure 3. The principle 'It is not the case that' is selected

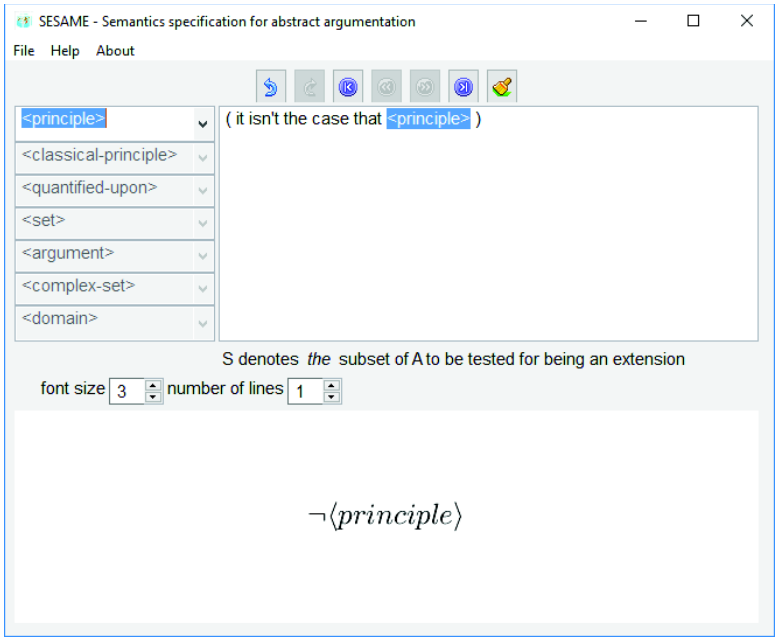


Figure 4. After a principle has been selected, the corresponding formula (lower white frame) is updated

For example, "it is not the case that ⟨principle⟩" gives rise to the formula ¬ ⟨principle⟩ (see Figure 4) and "⟨principle⟩ and ⟨principle⟩" gives rise to ⟨principle⟩ ∧ ⟨principle⟩.

Assume for example that the current non-terminal is ⟨classical-principle⟩. As the user clicks upon the development button labelled ⟨classical-principle⟩, a scroll menu is displayed (see Figure 7) which indicates that ⟨classical-principle⟩ can be replaced by "⟨set⟩ is conflict-free" or "⟨set⟩ satisfies admissibility" or "⟨set⟩ satisfies reinstatement". After choosing one of these three options, the user will eventually have to specify ⟨set⟩ by clicking upon the development button labelled ⟨set⟩ (see Figure 2) *etc.*

3.1. Grammar

The range of argumentation semantics that the user can specify is given by means of the grammar below. Moreover, the way the user can specify an argumentation semantics is constrained as discussed in Section 2: The grammar only accepts primitive statements of (non-)membership in X for some variable X denoting a set of arguments. An example is

for all a and b such that a$\mathfrak{R}$b, if a is in S then b is not in S.

A counter-example is the *same* semantics, when *expressed* as follows:

for all a in S, if b is in S then a$\mathfrak{R}$b fails.

The latter does not conform with the grammar since it resorts to a statement (*i.e.* $a\mathfrak{R}b$) that fails to be a membership statement as required.[5] Let us repeat that all the primitive

[5]Even when $a\mathfrak{R}b$ is expressed as $b \in \mathfrak{R}^+(a)$ or as $a \in \mathfrak{R}^-(b)$ because neither $\mathfrak{R}^+(a)$ nor $\mathfrak{R}^-(b)$ is a variable.

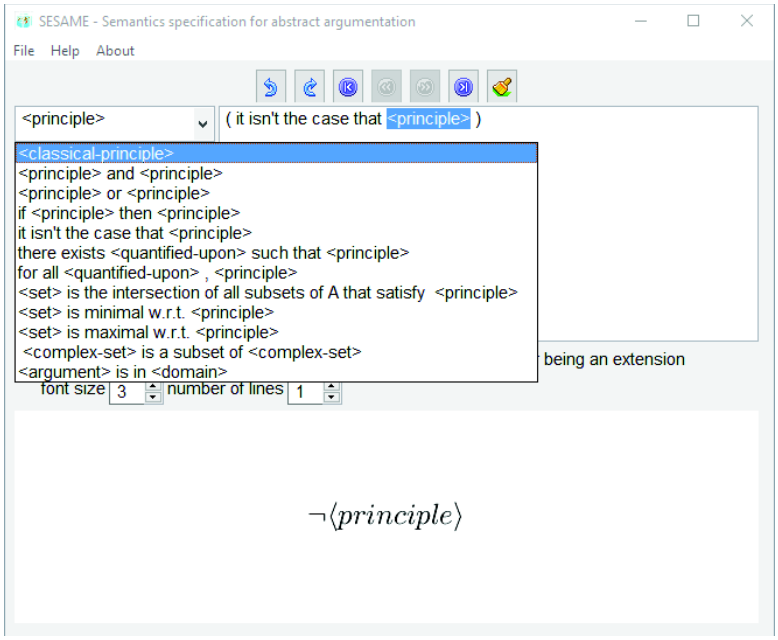


Figure 5. The principle 'classical-principle' is selected

statements to be encoded must be of the kind *a is in X* where *a* is a *variable* referring to an argument and *X* is either $\mathscr{A}$ or *S* or a *variable* referring to a subset of the arguments.

⟨*semantics*⟩ ::= ⟨*principle*⟩
⟨*principle*⟩ ::= ⟨*classical-principle*⟩
| ¬ ⟨*principle*⟩
| ⟨*principle*⟩ ⟨*operator*⟩ ⟨*principle*⟩
| ⟨*attack-quantification*⟩ ⟨*principle*⟩
| ⟨*membership-quantification*⟩ ⟨*principle*⟩
| ⟨*inclusion-quantification*⟩ ⟨*principle*⟩
| ⟨*argument*⟩ ∈ ⟨*set*⟩
⟨*classical-principle*⟩ ::= ⟨*set*⟩ is conflict-free
| ⟨*set*⟩ satisfies admissibility
| ⟨*set*⟩ satisfies reinstatement
⟨*attack-quantification*⟩ ::= ⟨*quantifier*⟩ ⟨*argument*⟩ $\Re$ ⟨*argument*⟩
⟨*membership-quantification*⟩ ::= ⟨*quantifier*⟩ ⟨*argument*⟩ ∈ ⟨*set*⟩
⟨*inclusion-quantification*⟩ ::= ⟨*quantifier*⟩ ⟨*set*⟩ ⊆ ⟨*set*⟩
⟨*operator*⟩ ::= ∧ | ∨ | →
⟨*quantifier*⟩ ::= ⋀ | ⋁
⟨*set*⟩ ::= $\mathscr{A}$ | S | S_1 | S_2 | . . .
⟨*argument*⟩ ::= a | b | c | . . . | x | y | z

$\mathscr{A}$ and *S* are parameters, all the other non-logical terminals in the grammar are *variables*: a, b,. . . , y, z, S_1, S_2,. . . In particular, a, b, . . . , y, z is *not* a list of arguments of $\mathscr{A}$.

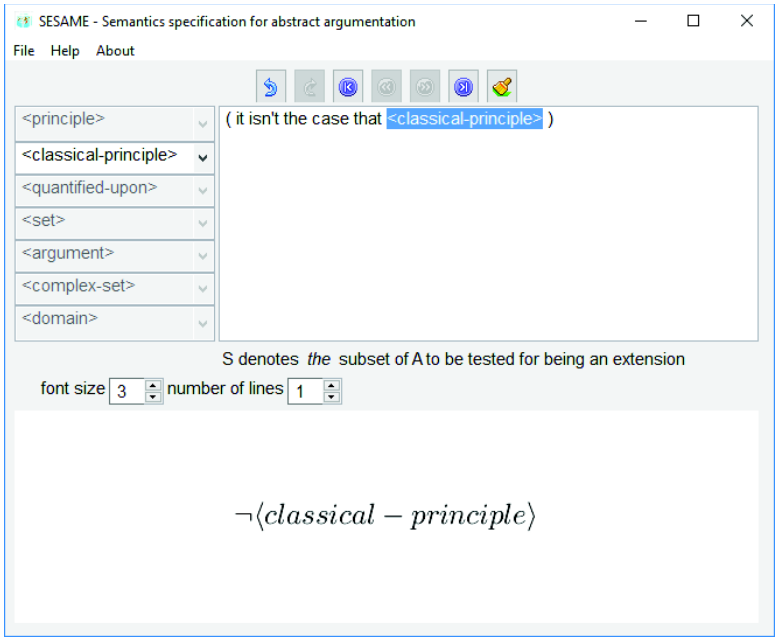
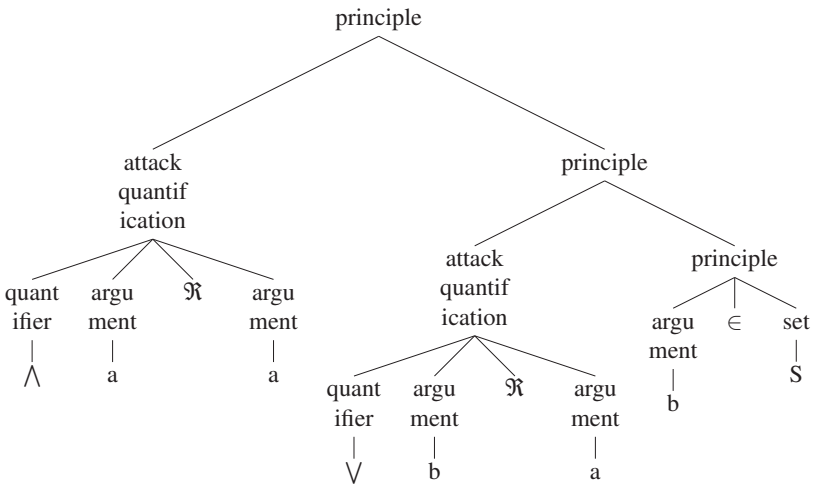


Figure 6. A classical-principle must be chosen

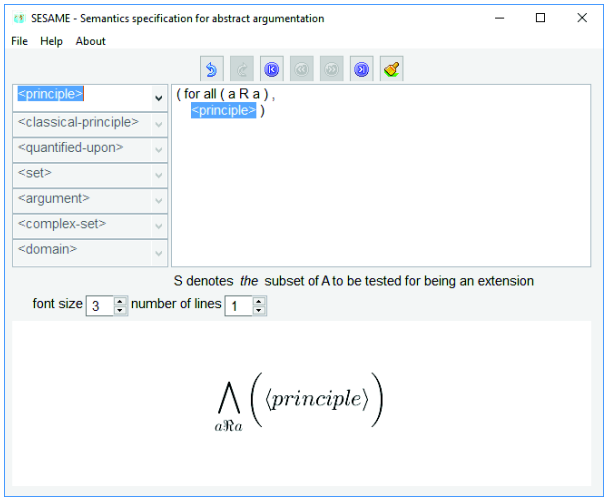
The user is to input her argumentation semantics by clicking upon development buttons so as to construct a decomposition tree for the grammar. Assume e.g. that the user is to define the principle that every self-attacking argument is attacked by an argument of S.



This decomposition tree (not visible by the user!) produces the formula $\bigwedge_{a\Re a}\ \bigvee_{b\Re a}\varphi_{(b\in S)}$ and corresponds to an appropriate sequence of clicking upon development buttons:

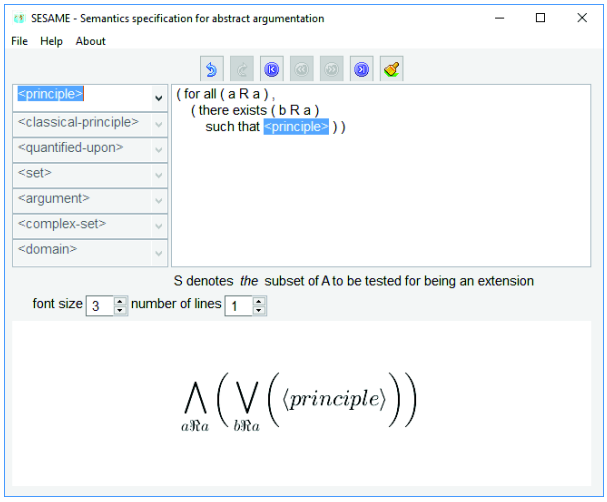
	development button		*choice from the scroll menu*
1.	⟨principle⟩	→	for all ⟨quantified-upon⟩, ⟨principle⟩
2.	⟨quantified-upon⟩	→	⟨argument⟩ $\mathfrak{R}$ ⟨argument⟩
3.	⟨argument⟩	→	*a*
4.	⟨argument⟩	→	*a*

At this point the current output is:



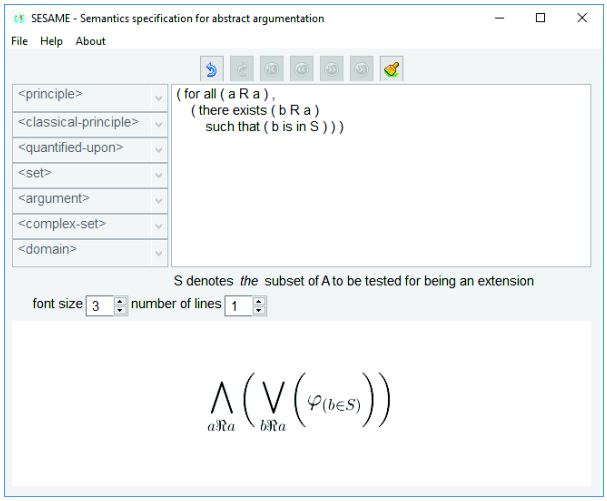
	development button		*choice from the scroll menu*
5.	⟨principle⟩	→	there exists ⟨quantified-upon⟩ such that ⟨principle⟩
6.	⟨quantified-upon⟩	→	⟨argument⟩ $\mathfrak{R}$ ⟨argument⟩
7.	⟨argument⟩	→	*b*
8.	⟨argument⟩	→	*a*

At this point the current output is:



	development button		*choice from the scroll menu*
9.	⟨principle⟩	→	⟨argument⟩ ∈ ⟨domain⟩
10.	⟨argument⟩	→	b
11.	⟨domain⟩	→	⟨set⟩
12.	⟨set⟩	→	S

At this point the output is final:



$\sigma_{(\mathscr{A},\mathfrak{R}),S}$ (where $\varphi_{(b\in S)}$ encodes the statement "b is in S", see footnote 2) is the formula:[6]

$$\bigwedge_{a\mathfrak{R}a}\bigvee_{b\mathfrak{R}a}\varphi_{(b\in S)} \wedge \bigwedge\Phi_S$$

4. Features of the System

4.1. Scroll Menus for Development Buttons

As development buttons correspond to non-terminals of the grammar, the items in the scroll menu attached to the button are the grammar rule(s) expanding the non-terminal. For instance, the scroll menu for ⟨argument⟩ displays the list a, b, ..., y, z and the scroll menu for ⟨set⟩ displays an initial series of the list S_1, S_2, ... Also, the scroll menu for ⟨classical-principle⟩ displays three items (see Figure 7).

As an exception, not all options in the scroll menu for ⟨principle⟩ are grammar rules expanding ⟨principle⟩. The extra options can be viewed as macros explicited in order to help the user. For instance, the grammar allows the user to specify that a subset of $\mathscr{A}$ is maximal for satisfying ⟨principle⟩. A casual user is not expected to figure out how to express this through the grammar. So, the scroll menu for ⟨principle⟩ includes the items:

[6] Let us repeat that $\bigwedge\Phi_S$ is not part of the user's specification. Moreover, following the generic notation, $\Phi_S = \{\varphi_{(a\in S)} \mid a \in S\} \cup \{\neg\varphi_{(a\in S)} \mid a \in \mathscr{A} \setminus S\}$.

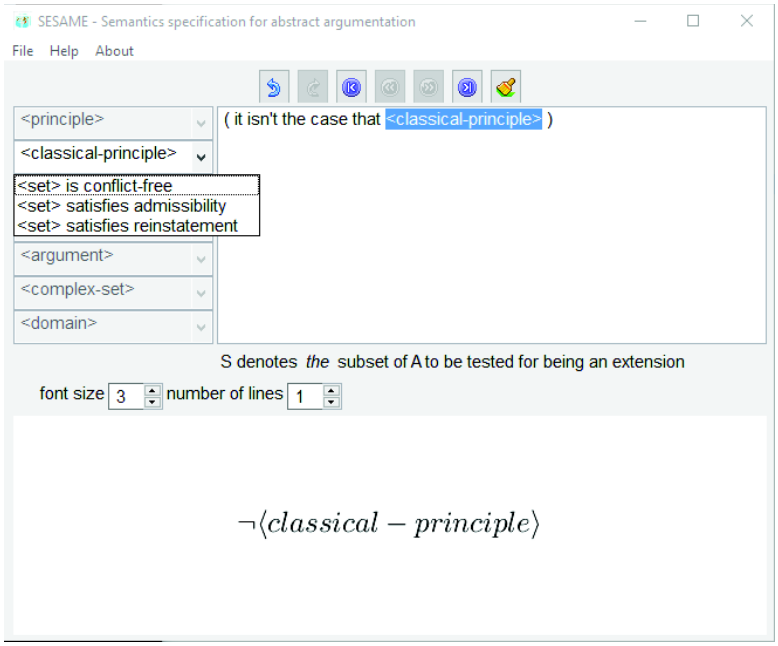


Figure 7. A classical-principle is being chosen

⟨set⟩ *is maximal wrt* ⟨principle⟩
⟨set⟩ *is minimal wrt* ⟨principle⟩
⟨set⟩ *is the intersection of all subsets of* $\mathscr{A}$ *that satisfy* ⟨principle⟩

The grammar allows the user to specify $\mathfrak{R}^+(a) \subseteq \mathfrak{R}^-(b) \cup (\mathscr{A} \setminus S)$ and other set-inclusion statements but only in a convoluted way. The system is to alleviate the burden on the user by providing her with appropriate items. Hence the scroll menu for ⟨principle⟩ includes:

⟨complex-set⟩ is a subset of ⟨complex-set⟩

This is why there is a development button labelled ⟨complex-set⟩ whose scroll menu is:

⟨domain⟩
$\mathscr{A}$ \ ⟨complex-set⟩
⟨complex-set⟩ ∪ ⟨complex-set⟩
⟨complex-set⟩ ∩ ⟨complex-set⟩

and there is a development button labelled ⟨domain⟩ whose scroll menu displays:

⟨set⟩
$\mathfrak{R}^+$⟨argument⟩
$\mathfrak{R}^-$⟨argumentt⟩

Also, the scroll menu for ⟨principle⟩ factors the three cases ⟨attack-quantification⟩, ⟨set-quantification⟩, and ⟨inclusion-quantification⟩ by means of the two items "for all ⟨quantified-upon⟩" and "there exists ⟨quantified-upon⟩" where ⟨quantified-upon⟩ is to be expanded to either ⟨argument⟩ $\mathfrak{R}$ ⟨argument⟩ or ⟨argument⟩ $\in$ ⟨set⟩ or ⟨set⟩ $\subseteq$ ⟨set⟩.

4.2. Current Non-terminal: Highlighted Development Button

At any time, the non-terminal to be developed next is highlighted (blue inverse video). By default, the order in which the non-terminals are developed is depth-first, left to right. However, the browsing buttons (see Figure 8) allow the user to override the default order and to freely choose the next non-terminal to be developed. Also, depending on which non-terminal is highlighted, a development button is enabled but the other development buttons are disabled (shadow display).

Figure 8. The buttons allowing to browse among non-terminals of the main white frame

4.3. The 'undo' and 'redo' Buttons

Immediately above the main white frame there is a series of buttons, of which the left-most two are *undo* and *redo* (see Figure 9). They allow the user to navigate through any step between the current state of the decomposition tree and its initial state. Of course, *undo* goes backward one step and *redo* goes forward one step (up to the current state).

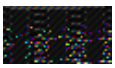


Figure 9. The 'undo' and 'redo' buttons

4.4. Output Files Menu

The *File* button offers a pull-down menu with an import item and various export options.

The first option stores an output file .txt with the content of the main white frame (that is, an expression of the user's argumentation semantics in semi-natural language). The second option stores an output file .tex with the LaTeX source of the content of the lower main frame (i.e., the formula $\sigma_{(\mathscr{A},\mathfrak{R}),S}$ in LaTeX). The third option stores an output file .jpg of the image of the LaTeX display of $\sigma_{(\mathscr{A},\mathfrak{R}),S}$. An option stores an output file .sesame with $\sigma_{(\mathscr{A},\mathfrak{R}),S}$ in internal format —that the *import* item of the menu downloads.

Lastly, there is an option that stores an output file .touistl of $\sigma_{(\mathscr{A},\mathfrak{R}),S}$ in the format of TouIST [7] which is a front-end for various SAT solvers. Indeed, a future version of our system will have a procedure permitting the user to input an argumentation graph $(\mathscr{A},\mathfrak{R})$ and $S \subseteq \mathscr{A}$ (also possibly varying $(\mathscr{A},\mathfrak{R})$ and S while still considering the same semantics σ) so that $\sigma_{(\mathscr{A},\mathfrak{R}),S}$ gets instantiated by $\mathscr{A},\mathfrak{R},S$ and fed to a SAT solver, thereby automatically informing the user whether the set S is a σ-extension of $(\mathscr{A},\mathfrak{R})$.

4.5. A Semantics ... or not?

The system achieves syntactical checking as it ensures that the specification entered by the user conforms with the grammar (e.g., the system makes it impossible that the input involves a statement of the form "the subset X of $\mathscr{A}$ is in the argument a"). Yet, there is no semantical check and it is largely possible for the user to input some dubious semantics (e.g., a "semantics" requiring that all self-attacking arguments are in S).

5. Conclusion and Perspectives

We have shortly described SESAME, a system (see www.irit.fr/SESAME) implemented in Java. It allows the user (guided by a simple BNF grammar) to specify an argumentation semantics σ, for which the system generates a propositional formula $\sigma_{(\mathscr{A},\mathfrak{R}),S}$ that turns out to be satisfiable if and only if S is a σ-extension of the argumentation graph $(\mathscr{A},\mathfrak{R})$. Such an argumentation semantics must conform to the grammar, and this may require some *ingenuity* from the user. The reason is that there are usually many ways to specify an argumentation semantics but the user must find out how to express hers in a way accepted by the grammar. To some extent, the system alleviates the burden on the user by allowing the user not to conform strictly to the *rules* of the grammar (but the system ensures that the user conforms with the *language* generated by the grammar).

When instantiated by a specific argumentation graph $(\mathscr{A},\mathfrak{R})$ and set S, the formula generated by SESAME may be fed to a SAT solver. Of course, such an approach is not meant to compete with optimized systems e.g. from ICCMA [8]. However, the ICCMA argumentation graphs benchmark may provide interesting instantiations allowing new semantics to be tested on a wide and diverse range of small argumentation graphs.

Moreover, we plan to extend the functionalities of the system so that, for example, recursive semantics of various kinds [9] can be specified (as to now, classical semantics captured by SESAME include grounded, complete, preferred, semi-stable, stage, ideal).

Acknowledgements. This work benefited from the support of the AMANDE project (ANR-13-BS02-0004) of the French National Research Agency (ANR).

References

[1] Philippe Besnard and Sylvie Doutre, Checking the Acceptability of a Set of Arguments, *Proc. of the 10th Int. Workshop on Non-Monotonic Reasoning (NMR'04)*, James Delgrande and Torsten Schaub (editors), pp. 59–64, Whistler, B.C., Canada, 2004.

[2] Michal Walicki and Sjur Dyrkolbotn, Finding Kernels or Solving SAT, *Discrete Algorithms*, **10** (2012), 146–164.

[3] Philippe Besnard, Sylvie Doutre, and Andreas Herzig, Encoding Argument Graphs in Logic, *Proc. of the 15th Int. Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'14)*, Anne Laurent, Olivier Strauss, Bernadette Bouchon-Meunier and Ronald Yager (editors), volume 443 of Communications in Computer and Information Science, pp. 345–354 (Part II), Montpellier, France, 2014.

[4] Uwe Egly, Sarah Gaggl, and Stefan Woltran, Answer-Set Programming Encodings for Argumentation Frameworks, *Argument and Computation*, **1**, (2010), 147–177.

[5] Phan Minh Dung, On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-person Games, *Artificial Intelligence*, **77** (1995), 321–357.

[6] Pietro Baroni and Massimiliano Giacomin, On Principle-based Evaluation of Extension-based Argumentation Semantics, *Artificial Intelligence*, **171**, (2007), 675–700.

[7] Khaled Skander Ben Slimane, Alexis Comte, Olivier Gasquet, Abdelwahab Heba, Olivier Lezaud, Frédéric Maris, and Maël Valais, Twist your Logic with TouIST, *Proc. of the 4th Int. Congress on Tools for Teaching Logic (TTL'15)*, Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat and François Schwarzentruber (editors), Rennes, France, 2015.

[8] Matthias Thimm and Serena Villata (editors), *Systems Descriptions of the 1st Int. Competition on Computational Models of Argumentation (ICCMA'15)*, 2015. ArXiv:1510.05373.

[9] Pietro Baroni and Massimiliano Giacomin, Semantics of Abstract Argument Systems, in *Argumentation in Artificial Intelligence*, Guillermo Simari and Iyad Rahwan (editors), pp. 25–44, 2009, Springer.

[10] Günther Charwat, Wolfgang Dvorák, Sarah Alice Gaggl, Johannes Peter Wallner, Stefan Woltran, Methods for Solving Reasoning Problems in Abstract Argumentation –A Survey, *Artificial Intelligence*, **220**, (2015), 28-63.