An Algorithm for Stage Semantics

Martin CAMINADA [a]
[a] *University of Luxembourg*

Abstract. In the current paper, we re-examine the concept of stage semantics, which is one of the oldest semantics for abstract argumentation. Using a formal treatment of its properties, we explain how the intuition behind stage semantics differs from the intuition behind the admissibility based semantics that most scholars in argumentation theory are familiar with. We then provide a labelling-based algorithm for computing all stage extensions, based on earlier algorithms for computing all preferred, stable and semi-stable extensions.

1. Introduction

The concept of stage semantics for abstract argumentation was first introduced by Verheij [15] and has subsequently been worked out in Verheij's DEFLOG system [16,17], which can be regarded as a generalization of the abstract argumentation theory of Dung [10]. Although stage semantics is one of the oldest semantics for abstract argumentation, it has so far remained relatively unknown, which might have to do with the fact that it was originally stated not in terms of the usual extensions approach, but in the form of pairs (J, D) where J is a set of justified arguments and D is a set of defeated arguments [15]. Nevertheless, there exist good reasons for treating stage semantics as one of the mainstream semantics for abstract argumentation, not only because it can be expressed using a relatively simple and elegant principle, but also because it implements a fundamentally different intuition than the traditional admissibility based semantics (such as complete, grounded and preferred [10], ideal [11] or semi-stable [15,5]).

Despite of the differences between stage semantics and the traditional admissibility-based semantics, it is still possible to provide an algorithm for computing all stage extensions, that is very close to previously stated algorithms for computing all preferred, stable and semi-stable extensions [6,14], as is demonstrated in the current paper.

2. Stage Semantics

In Verheij's original work [15] stage semantics was defined in terms of pairs of sets of arguments. In the current paper, however, we will describe stage semantics in terms of the more commonly applied extensions approach. We assume familiarity with basic argumentation concepts, such as that of an argumentation framework, conflict-free sets, admissible sets, complete extensions, preferred extensions, stable extensions and the grounded extension. Definitions of these can be found in [10]. In the current paper, we only consider finite argumentation frameworks.

If A is an argument then we write A^+ for the set of arguments attacked by A. Similarly, if $\mathcal{A}rgs$ is a set of arguments then we write $\mathcal{A}rgs^+$ to refer to the set of arguments attacked by at least one argument in $\mathcal{A}rgs$.

Definition 1. *Let $AF = (Ar, att)$ be an argumentation framework. A stage extension is a conflict-free set $\mathcal{A}rgs \subseteq Ar$ where $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is maximal (w.r.t. set inclusion) among all conflict-free sets.*

Stage semantics can to some extent be compared to semi-stable semantics, which is essentially an admissible set $\mathcal{A}rgs$ where $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is maximal. In the remainder of this paper, we refer to $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ as the *range* of $\mathcal{A}rgs$, a term that was first introduced in [15]. Thus, where semi-stable extensions are admissible sets with maximal range, stage extensions are conflict-free sets with maximal range.

One could examine whether the same principle can also be applied to other semantics. That is, what if for instance one would look at complete, preferred or stable extensions with maximal range? It turns out that doing so does not yield any additional semantics. Complete extensions with maximal range, as well as preferred extensions with maximal range, are semi-stable extensions.[1] Stable extensions by definition have a maximal range, so selecting the stable extensions with maximal range simply means selecting all stable extensions. An overview of the effects of selecting sets and extensions with maximal range is provided in Table 1.

input extensions/sets	conflict-free sets	admissible sets	complete extensions	preferred extensions	stable extensions
result when selecting for maximal range	stage extensions	semi-stable extensions	semi-stable extensions	semi-stable extensions	stable extensions

Table 1. Selecting the extensions with a maximal range, given a semantics.

As an example of how stage semantics operates, consider the first argumentation framework on the left of Figure 1. Here, there exist five conflict-free sets: $\emptyset$, $\{A\}$, $\{B\}$, $\{C\}$ and $\{A, C\}$. Both $\{B\}$ and $\{A, C\}$ are maximal, but only $\{A, C\}$ has a maximal range, so only $\{A, C\}$ is a stage extension. This illustrates that selecting a maximal conflict-free set is different than selecting a conflict-free set with maximal range.[2]

Another example is the second argumentation framework of Figure 1. Here, there exist pricisely two stage extensions: $\{A, D\}$ and $\{B, D\}$ (both of which have a range of $\{A, B, C.D\}$). Hence, the results of stage semantics of this example are in line with the results of more established semantics like preferred, stable and semi-stable.

In the third argumentation framework of Figure 1, the two-cycle of the previous example has been replaced by a three-cycle. Here, there exist three stage extensions $\{A, E\}$, $\{B, E\}$ and $\{C, E\}$ (with corresponding ranges $\{A, B, D, E\}$, $\{B, C, D, E\}$ and $\{C, A, D, E\}$). This is in contrast with the admissibility based semantics (grounded, preferred, complete, ideal and semi-stable) which all yield $\emptyset$ as the only extension.

[1]The equivalence between admissible sets with maximal ranges and complete extensions with maximal ranges has been proved in [5], and the equivalence between complete extensions with maximal ranges and preferred extensions with maximal ranges can be proved in a similar way.

[2]Just like selecting a maximal admissible set (preferred) is different than selecting an admissible set with a maximal range (semi-stable).

Hence, one of the advantages of stage semantics is that odd and even loops are treated equally. The only other well-known semantics with this property is CF2 [3].[3]

The fourth argumentation framework of Figure 1 is where stage semantics yields a different result than obtained by the admissibility based semantics, as well as by CF2. The only stage extension here is $\{B\}$, where the admissibility based semantics all yield $\emptyset$ as the only extension.

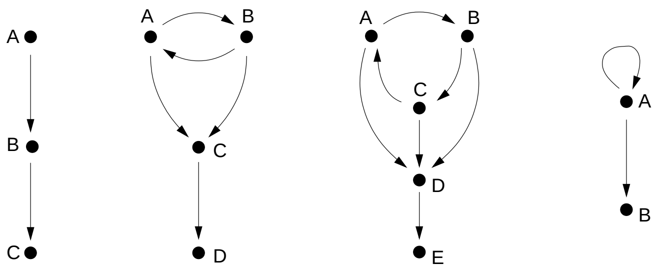

Figure 1. Four argumentation frameworks

It holds that every stable extension is a stage extension (just like every stable extension is a semi-stable extension [5]).

Theorem 1. *Let $\mathcal{A}rgs$ be a stable extension of argumentation framework (Ar, att). $\mathcal{A}rgs$ is also a stage extension of (Ar, att).*

Proof. Let $\mathcal{A}rgs$ be a stable extension of (Ar, att). Then $\mathcal{A}rgs$ is a conflict-free set that attacks every argument in $Ar \backslash \mathcal{A}rgs$. This means that $\mathcal{A}rgs \cup \mathcal{A}rgs^{+} = Ar$. Therefore, $\mathcal{A}rgs \cup \mathcal{A}rgs^{+}$ is maximal (it cannot be a proper superset of Ar). Therefore, $\mathcal{A}rgs$ is a stage extension. $\square$

It also holds that if there exists at least one stable extension, then every stage extension is also a stable extension (just like if there exists at least one stable extension, then every semi-stable extension is also a stable extension [5]).

Theorem 2. *Let (Ar, att) be an argumentation framework that has at least one stable extension. It then holds that every stage extension is also a stable extension.*

Proof. Let $\mathcal{A}rgs$ be a stable extension of (Ar, att). It then holds that $\mathcal{A}rgs$ is a conflict-free set with $\mathcal{A}rgs \cup \mathcal{A}rgs^{+} = Ar$. Therefore, every stage extension $\mathcal{A}rgs'$ will have to satisfy $\mathcal{A}rgs' \cup \mathcal{A}rgs'^{+} = Ar$ in order to have a maximal range. This means that every stage extension will also be a stable extension. $\square$

Theorem 1 and 2 can be seen as special instances of the results obtained in DEFLOG [16,17]. Apart from the extensions approach, it is also possible to describe stage semantics in terms of argument labellings [4,8,7].

[3]CF2 semantics does have the advantage that the grounded extension is a subset of *every* CF2 extension, whereas for stage semantics one has the weaker property that the grounded extension is a subset of *at least one* stage extension. See [7] for more details.

Definition 2. *Let (Ar, att) be an argumentation framework. A labelling is a function $\mathcal{L}ab : Ar \to \{\mathtt{in}, \mathtt{out}, \mathtt{undec}\}$.*

If $\mathcal{L}ab$ is a labelling then we write $\mathtt{in}(\mathcal{L}ab)$ for $\{A \mid \mathcal{L}ab(A) = \mathtt{in}\}$, $\mathtt{out}(\mathcal{L}ab)$ for $\{A \mid \mathcal{L}ab(A) = \mathtt{out}\}$ and $\mathtt{undec}(\mathcal{L}ab)$ for $\{A \mid \mathcal{L}ab(A) = \mathtt{undec}\}$. Since a labelling is a function, it can be represented as a set of pairs. In this paper we will sometimes use an alternative way to represent a labelling, as a partition $(\mathtt{in}(\mathcal{L}ab), \mathtt{out}(\mathcal{L}ab), \mathtt{undec}(\mathcal{L}ab))$.

Definition 3. *Let (Ar, att) be an argumentation framework. A conflict-free labelling is a labelling such that for every $A \in Ar$ it holds that:*

1. *if $\mathcal{L}ab(A) = \mathtt{in}$ then $\forall B \in Ar : (B att A \supset \mathcal{L}ab(B) \neq \mathtt{in})$*
2. *if $\mathcal{L}ab(A) = \mathtt{out}$ then $\exists B \in Ar : (B att A \wedge \mathcal{L}ab(B) = \mathtt{in})$*

The definition of a conflict-free labelling is almost equal to that of an admissible labelling in [8]. The only difference is that for an admissible labelling, the first clause is "if $\mathcal{L}ab(A) = \mathtt{in}$ then $\forall B \in Ar : (B att A \supset \mathcal{L}ab(B) = \mathtt{out})$", which is stronger than the first clause of Definition 3. It immediately follows that every admissible labelling is also a conflict-free labelling, just like every admissible set is also a conflict-free set.

Definition 4. *Let (Ar, att) be an argumentation framework. $\mathcal{L}ab$ is a stage labelling iff $\mathcal{L}ab$ is a conflict-free labelling where* `undec` *is minimal (w.r.t. set inclusion) among all conflict-free labellings.*

It can be verified that stage extensions and stage labellings stand in a one-to-one relationship to each other.

Proposition 1. *Let $AF = (Ar, att)$ be an argumentation framework.*

1. *If $\mathcal{A}rgs$ is a stage extension of AF then $\mathtt{Ext2Lab}(\mathcal{A}rgs)$ is a stage labelling of AF, where $\mathtt{Ext2Lab}(\mathcal{A}rgs) = (\mathcal{A}rgs, \mathcal{A}rgs^+, Ar \backslash (\mathcal{A}rgs \cup \mathcal{A}rgs^+))$.*
2. *If $\mathcal{L}ab$ is a stage labelling of AF then $\mathtt{Lab2Ext}(\mathcal{L}ab)$ is a stage extension of AF, where $\mathtt{Lab2Ext}(\mathcal{L}ab) = \mathtt{in}(\mathcal{L}ab)$.*

Moreover, when restricted to stage extensions and stage labellings, the functions `Ext2Lab` *and* `Lab2Ext` *become bijective and each other's inverses.*

The proof of Proposition 1 is included in [7], where it is also proved that every stable labelling (in the sense of [4,8]) is also a stage labelling, and that if there exists at least one stable labelling, then every stage labelling is also a stable labelling.

3. An Algorithm

In the current section we provide an algorithm for computing all stage labellings of a given argumentation framework. Using the result of Proposition 1 we then also obtain the stage extensions of the argumentation framework, which are after all simply the sets of `in`-labelled arguments of the stage labellings.

The basic idea of the algorithm is to start with a labelling in which every argument is labelled `in`. This labelling will subsequently be referred to as the *all*-`in` labelling. Then, a sequence of *transition steps* is applied, where each step resolves a conflict between two `in`-labelled arguments where one attacks the other. The algorithm described in the current section is a slightly modified version of the earlier developed algorithm for computing all semi-stable labellings[4] [6]. The main difference is that where the semi-stable algorithm is based on the notion of an admissible labelling, the currently presented stage algorithm is based on the notion of a conflict-free labelling.

Definition 5. *Let $\mathcal{L}ab$ be a labelling of argumentation framework $AF = (Ar, att)$.*

1. *an* `in`*-labelled argument A is called* illegally `in` *iff* $\exists B \in Ar : (B att A \wedge \mathcal{L}ab(B) = \texttt{in}) \vee \exists B \in Ar : (A att B \wedge \mathcal{L}ab(B) = \texttt{in})$
2. *an* `out`*-labelled argument A is called* illegally `out` *iff* $\neg \exists B \in Ar : (B att A \wedge \mathcal{L}ab(B) = \texttt{in})$

In essence, an argument is illegally `in` iff it violates point 1 of Definition 3 and is illegally `out` iff it violates point 2 of Definition 3. It then follows that a labelling is conflict-free iff it does not have any argument that is illegally `in` or illegally `out`.

Definition 6. *Let $\mathcal{L}ab$ be a labelling of argumentation framework (Ar, att) and $A \in Ar$ and argument that is illegally* `in` *in $\mathcal{L}ab$. A* transition step *on A in $\mathcal{L}ab$ consists of the following:*

1. *the label of A is changed from* `in` *to* `out`
2. *for every $B \in \{A\} \cup A^+$, if B is illegally* `out` *then the label of B is changed from* `out` *to* `undec`

It can be observed that each transition step preserves the absence of arguments that are illegally `out`. That is, if a labelling does not have any argument that is illegally `out` before a transition step, then there will still be no argument that is illegally `out` after the transition step. Moreover, since each transition step reduces the number of illegally `in`-labelled arguments by at least one, subsequently applying transition steps starting from the all-`in` labelling ultimately yields a labelling without any illegally `in`-labelled arguments.

Definition 7. *A* transition sequence *is a list $[\mathcal{L}ab_0, A_1, \mathcal{L}ab_1, A_2, \mathcal{L}ab_2, \ldots, A_n, \mathcal{L}ab_n]$ ($n \geq 0$) where $\mathcal{L}ab_0$ is the all-*`in` *labelling, each A_i ($1 \leq i \leq n$) is an argument that is illegally* `in` *in $\mathcal{L}ab_{i-1}$, and every $\mathcal{L}ab_i$ ($1 \leq i \leq n$) is the result of doing a transition step on A_i in $\mathcal{L}ab_{i-1}$. A transition sequence is called* terminated *iff $\mathcal{L}ab_n$ does not have any argument that is illegally* `in`*.*

Since we only consider finite argumentation frameworks, it holds that every transition sequence can be extended to a terminated transition sequence, in which a finite number of transitions have been performed. We say that a transition sequence *yields $\mathcal{L}ab$* if its last labelling is $\mathcal{L}ab$.

As an example of a transition sequence, consider the third argumentation framework of Figure 1. Starting with the all-`in` labelling $\mathcal{L}ab_0 = (\{A, B, C, D\}, \emptyset, \emptyset)$, one could, for instance, select argument A to do a transition step on, resulting in the labelling $\mathcal{L}ab_1 = (\{B, C, D, E\}, \{A\}, \emptyset)$. Subsequently one could, for instance, select argument

[4]Other variations of the same algorithm exist for computing all preferred and all stable labellings [6].

C to do a transition step on. Once C is relabelled `out`, A becomes illegally `out` and is therefore relabelled `undec` in the same transition step. Hence the resulting labelling $\mathcal{L}ab_2$ is $(\{B, D, E\}, \{C\}, \{A\})$. Subsequently, one could select argument D to do a transtion step on, resulting in the labelling $\mathcal{L}ab_3 = (\{B, E\}, \{C, D\}, \{A\})$. This labelling does not have any argument that is illegaly `in`, so the transition sequence is terminated. The overall transition sequence is $[\mathcal{L}ab_0, A, \mathcal{L}ab_1, C, \mathcal{L}ab_2, D, \mathcal{L}ab_3]$.

Lemma 1. *Let $[\mathcal{L}ab_0, A_1, \mathcal{L}ab_1, \ldots, A_n, \mathcal{L}ab_n]$ be a terminated transition sequence. It holds that $\mathcal{L}ab_n$ is a conflict-free labelling.*

Proof. Since the all-`in` labelling does not have any argument that is illegally `out` (it has no `out`-labelled argument at all) and each transition step preserves the absence of illegally `out`-labelled arguments, it follows that each terminated transition sequence yields a labelling without any illegally `out`-labelled arguments. Since a terminated transition sequence also yields a labelling without any illegally `in`-labelled arguments (otherwise the transition sequence would not be terminated), it follows that the result yielded by a terminated transition sequence is a conflict-free labelling. □

As an example, consider the argumentation framework on the left of Figure 1. An example of a (terminated) transition sequence is, starting from the all-`in` labelling, first to perform a transition step on B, resulting in a labelling $(\{A, C\}, \{B\}, \emptyset)$, which is conflict-free, thus terminating the transition sequence. Another possibility would be first to perform a transition step on A, resulting in a labelling $(\{B, C\}, \emptyset, \{A\})$, and subsequently performing transition steps on B or C, resulting in the conflict-free labellings $(\{C\}, \emptyset, \{A, B\})$ and $(\{B\}, \{C\}, \{A\})$, respectively. Other transition sequences also exist. Only the first obtained labelling is a stage labelling. All other labellings that are yielded by terminated transition sequences are conflict-free but not stage. Fortunately, it holds that, as a general rule, every stage labelling is included in the results yielded by the terminated transition sequences, as is expressed by the following theorem.

Theorem 3. *Let $\mathcal{L}ab_{stage}$ be a stage labelling of argumentation framework (Ar, att). There exists a terminated transition sequence which yields $\mathcal{L}ab_{stage}$.*

Proof. We prove the theorem by, given a stage labelling, constructing a terminated transition sequence that yields it. The idea is to construct this transition sequence in two phases, first by doing transition steps on the arguments that are labelled `out` by $\mathcal{L}ab$, then by doing transition steps on the arguments that are labelled `undec` by $\mathcal{L}ab$.
Let $\mathcal{L}ab_{stage}$ be a stage labelling. Let $[\mathcal{L}ab_0, A_1, \mathcal{L}ab_1, \ldots, A_m, \mathcal{L}ab_m]$ $(m \geq 0)$ be a (possibly unterminated) transition sequence where $A_1, \ldots, A_m$ are the arguments that are labelled `out` by $\mathcal{L}ab_{stage}$ (that is: $\{A_1, \ldots, A_m\} = \texttt{out}(\mathcal{L}ab_{stage})$. This is a correct transition sequence, because every A_i $(1 \leq i \leq m)$ will be illegally `in` until it is relabelled to `out` (this is because every A_i is legally `out` in $\mathcal{L}ab_{stage}$, so it has an attacker that is labelled `in` by $\mathcal{L}ab_{stage}$, and since we do not relabel this attacker in any of the transition steps, it will be labelled `in` by every $\mathcal{L}ab_i$ $(0 \leq i \leq m)$). Furthermore, none of the transition steps will relabel any `out`-labelled argument to `undec` because every `out`-labelled argument will stay legally `out` throughout the transition sequence. This implies that $\texttt{undec}(\mathcal{L}ab_m) = \emptyset$. Furthermore, since transition steps are done on every argument that is labelled `out` by $\mathcal{L}ab_{stage}$, it follows that $\texttt{out}(\mathcal{L}ab_m) = \texttt{out}(\mathcal{L}ab_{stage})$. Since no transition steps are done on arguments that are labelled `in` by $\mathcal{L}ab_{stage}$, it holds that

$\texttt{in}(\mathcal{L}ab_m) \supseteq \texttt{in}(\mathcal{L}ab_{stage})$.
We now continue the transition sequence, this time by doing transition steps not on arguments that are labeled `out` by $\mathcal{L}ab_{stage}$, but on arguments that are labelled `undec` by $\mathcal{L}ab_{stage}$. The idea is to keep doing this until there are no more arguments in $\texttt{undec}(\mathcal{L}ab_{stage})$ that are illegally `in` in the transition sequence. Thus, the extended transition sequence becomes $[\mathcal{L}ab_0, A_1, \mathcal{L}ab_1, \ldots, A_m, \mathcal{L}ab_m, A_{m+1}, \mathcal{L}ab_{m+1}, \ldots, A_{m+n}, \mathcal{L}ab_{m+n}]$ ($m, n \geq 0$) where:
(1) $\{A_{m+1}, \ldots, A_{m+n}\} \subseteq \texttt{undec}(\mathcal{L}ab_{stage})$, and
(2) $\mathcal{L}ab_{m+n}$ does not have any illegally `in`-labelled argument that is an element of $\texttt{undec}(\mathcal{L}ab_{stage})$.
Not only does $\mathcal{L}ab_{m+n}$ not have any illegally `in`-labelled argument that is an element of $\texttt{undec}(\mathcal{L}ab_{stage})$, it also does not have any illegally `in`-labelled argument that is an element of $\texttt{out}(\mathcal{L}ab_{stage})$ (this is because these arguments have been relabelled to `out` in the first part of the transition sequence), and it also does not have any illegally `in`-labelled argument that is an element of $\texttt{in}(\mathcal{L}ab_{stage})$ (this is because if there exists such an argument (say A), then it must have a conflict with an `in`-labelled argument from $\texttt{undec}(\mathcal{L}ab_{stage})$ (say B), but then it follows that B is also illegally `in` in $\mathcal{L}ab_{m+n}$: contradiction). From the fact that $\mathcal{L}ab_{m+n}$ does not have any illegally `in`-labelled argument that is in $\texttt{undec}(\mathcal{L}ab_{stage})$, in $\texttt{out}(\mathcal{L}ab_{stage})$ or in $\texttt{in}(\mathcal{L}ab_{stage})$, it holds that $\mathcal{L}ab_{m+n}$ does not have any illegally `in`-labelled argument at all. This means that the transition sequence is terminated, and that therefore (Lemma 1) $\mathcal{L}ab_{m+n}$ is a conflict-free labelling. Since no transition steps on any arguments in $\texttt{in}(\mathcal{L}ab_{stage})$ were performed, it holds that $\texttt{in}(\mathcal{L}ab_{m+n}) \supseteq \texttt{in}(\mathcal{L}ab_{stage})$. Furthermore, since $\texttt{out}(\mathcal{L}ab_m) = \texttt{out}(\mathcal{L}ab_{stage})$, and none of the transition steps following $\mathcal{L}ab_m$ relabels any of these `out`-labelled arguments to `undec` (they will always stay legally `out` because they all have attackers in $\texttt{in}(\mathcal{L}ab_{stage})$ that are not selected for any transition steps) it also holds that $\texttt{out}(\mathcal{L}ab_{m+n}) \supseteq \texttt{out}(\mathcal{L}ab_{stage})$. From the fact that $\texttt{in}(\mathcal{L}ab_{m+n}) \supseteq \texttt{in}(\mathcal{L}ab_{stage})$ and $\texttt{out}(\mathcal{L}ab_{m+n}) \supseteq \texttt{out}(\mathcal{L}ab_{stage})$ it follows that $\texttt{undec}(\mathcal{L}ab_{m+n}) \subseteq \texttt{undec}(\mathcal{L}ab_{stage})$. However, since $\mathcal{L}ab_{stage}$ is a stage labelling and $\mathcal{L}ab_{m+n}$ is a conflict-free labelling, it follows that $\texttt{undec}(\mathcal{L}ab_{m+n})$ cannot be a strict subset of $\mathcal{L}ab_{stage}$. Therefore, it must hold that $\texttt{undec}(\mathcal{L}ab_{m+n}) = \texttt{undec}(\mathcal{L}ab_{stage})$. It then follows that $\texttt{in}(\mathcal{L}ab_{m+n}) = \texttt{in}(\mathcal{L}ab_{stage})$ and $\texttt{out}(\mathcal{L}ab_{m+n}) = \texttt{out}(\mathcal{L}ab_{stage})$. This means that $\mathcal{L}ab_{m+n} = \mathcal{L}ab_{stage}$. □

Since each terminated transition sequence yields a labelling that is conflict-free, Theorem 3 allows for a simple way of obtaining all stage labellings: simply produce all terminated transition sequences and select the results with minimal `undec`. Fortunately, it is not necessary to compute *all* of the terminated transition sequences. This is because during the course of a transition sequence, the set of `undec`-labelled arguments either stays the same or increases, as is stated in the following proposition.

Proposition 2. *Let $[\mathcal{L}ab_0, A_1, \mathcal{L}ab_1, \ldots, A_n, \mathcal{L}ab_n]$ be a transition sequence. For any $1 \leq i \leq n$ it holds that* $\texttt{undec}(\mathcal{L}ab_{i-1}) \subseteq \texttt{undec}(\mathcal{L}ab_i)$.

Suppose we have already computed a terminated transition sequence yielding labelling $\mathcal{L}ab$, and we are currently computing a (not yet terminated) transition sequence whose last labelling is $\mathcal{L}ab_i$. If $\texttt{undec}(\mathcal{L}ab) \subsetneqq \texttt{undec}(\mathcal{L}ab_i)$ then Proposition 2 tells us that the current transition sequence will never yield a stage extension once it is terminated, and that it would therefore be a good idea to backtrack to another possibility. This allows us to prune the search space, a possibility that was not available in Verheij's original treatment of the dynamics of stage semantics [16,17].

Overall, the algorithm for computing the stage labellings of a given argumentation framework can be described as follows.

```
potential_stage_labs := $\emptyset$;   find_stage_labs(all-in);
print potential_stage_labs;   end;

procedure find_stage_labs($\mathcal{L}ab$)
  # if we found something that is worse than found earlier
  # then prune the search tree and backtrack
  if $\exists \mathcal{L}ab' \in$ potential_stage_labs: $\mathtt{undec}(\mathcal{L}ab') \subsetneq \mathtt{undec}(\mathcal{L}ab)$ then return;
  # now see if our current transition sequence has terminated
  if $\mathcal{L}ab$ does not have an argument that is illegally $\mathtt{in}$ then
    for each $\mathcal{L}ab' \subseteq$ potential_stage_labs
      # if an old candidate is worse than the new candidate: remove
      if $\mathtt{undec}(\mathcal{L}ab) \subsetneq \mathtt{undec}(\mathcal{L}ab')$ then
        potential_stage_labs := potential_stage_labs - $\{\mathcal{L}ab'\}$;
      endif;
    endfor;
    # add our newly found labelling as a candidate
    # we already know that it is not worse than what we already have
    potential_stage_labs := potential_stage_labs $\cup \{\mathcal{L}ab\}$;
    return; # we are done with this one; try next possibility
  else
    for each argument $A$ that is illegally $\mathtt{in}$ in $\mathcal{L}ab$
      find_stage_labs(transition_step($A$, $\mathcal{L}ab$));
    endfor;
  endif;
endproc;

procedure transition_step($A$, $\mathcal{L}ab$)
  $\mathcal{L}ab' := \mathcal{L}ab$;
  # relabel argument $A$ from in to out
  $\mathcal{L}ab'$ := $(\mathcal{L}ab' - \{(A, \mathtt{in})\}) \cup \{(A, \mathtt{out})\}$;
  # relabel any resulting illegal out to undec
  for each $B$ in $\{A\} \cup A^+$
    if $B$ is illegally out then $\mathcal{L}ab'$ := $(\mathcal{L}ab' - \{(B, \mathtt{out})\}) \cup \{(B, \mathtt{undec})\}$;
  endfor;
  return $\mathcal{L}ab'$;
endproc;
```

A software implementation of the above algorithm is presented at the COMMA demo session. Apart from computing the stage labellings (extensions), the software is also able to compute the grounded, preferred, stable and semi-stable labellings (extensions) using the algorithms described in [6,14].

4. Stage Semantics and Maximal Consistency

There exists an alternative way to describe the concept of stage semantics. In essence, what stage semantics does is taking a maximal subgraph of the argumentation framework that has at least one stable extension. A stage extension is then a stable extension of such a maximal subgraph. Similar observations have been made in the context of DEFLOG [16,17]. In the current section, however, we treat these results in the context of abstract argumentation.

Definition 8. *Let $AF = (Ar, att)$ be an argumentation framework and $\mathcal{A}rgs \subseteq Ar$. We define a* subframework $AF|_{\mathcal{A}rgs}$ *of AF as $(\mathcal{A}rgs, att \cap (\mathcal{A}rgs \times \mathcal{A}rgs))$. If $AF|_{\mathcal{A}rgs_1}$ and $AF|_{\mathcal{A}rgs_2}$ are subframeworks of AF then we say that $AF|_{\mathcal{A}rgs_2}$ is at least as big as $AF|_{\mathcal{A}rgs_1}$ iff $\mathcal{A}rgs_1 \subseteq \mathcal{A}rgs_2$.*

Proposition 3. *Let $AF = (Ar, att)$ be an argumentation framework and $\mathcal{A}rgs \subseteq Ar$. The following two statements are equivalent:*

1. *$\mathcal{A}rgs$ is a conflict-free set of AF*
2. *$\mathcal{A}rgs$ is a stable extension of $AF|_{\mathcal{A}rgs \cup \mathcal{A}rgs^+}$*

Theorem 4. *Let $AF = (Ar, att)$ be an argumentation framework and $\mathcal{A}rgs \subseteq Ar$. The following two statements are equivalent.*

1. *$\mathcal{A}rgs$ is a conflict-free set of AF where $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is maximal (w.r.t. set inclusion) among all conflict-free sets (that is, $\mathcal{A}rgs$ is a stage extension of AF).*
2. *$\mathcal{A}rgs$ is a stable extension of a maximal subframework of AF that has at least one stable extension.*

Proof. "from 1 to 2": Let $\mathcal{A}rgs$ be a conflict-free set of AF where $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is maximal. From Proposition 3 it follows that $\mathcal{A}rgs$ is a stable extension of $AF|_{\mathcal{A}rgs \cup \mathcal{A}rgs^+}$. So $\mathcal{A}rgs$ is a stable extension of a subframework of AF that has at least one stable extension. We now prove that $AF|_{\mathcal{A}rgs \cup \mathcal{A}rgs^+}$ is also a *maximal* subframework that has at least one stable extension. Let $\mathcal{A}rgs'_{range} \supsetneqq \mathcal{A}rgs \cup \mathcal{A}rgs^+$ be such that $AF|_{\mathcal{A}rgs'_{range}}$ has at least one stable extension, and let $\mathcal{A}rgs'$ be such a stable extension. It then follows that $\mathcal{A}rgs' \cup \mathcal{A}rgs'^+ = \mathcal{A}rgs'_{range}$. From Proposition 3 it then follows that $\mathcal{A}rgs'$ is a conflict-free set of AF. However, since $\mathcal{A}rgs' \cup \mathcal{A}rgs'^+ \supsetneqq \mathcal{A}rgs \cup \mathcal{A}rgs^+$, it follows that $\mathcal{A}rgs$ does not have a maximal range. Contradiction.
"from 2 to 1": Let $\mathcal{A}rgs$ be a stable extension of a maximal subframework that has at least one stable extension. It follows that this maximal subframework is $AF|_{\mathcal{A}rgs \cup \mathcal{A}rgs^+}$. Then from Proposition 3 it follows that $\mathcal{A}rgs$ is a conflict-free set of AF. We now prove that it is also a conflict-free set with a maximal range. Let $\mathcal{A}rgs'$ be a conflict-free set of AF such that $\mathcal{A}rgs \cup \mathcal{A}rgs^+ \subsetneqq \mathcal{A}rgs' \cup \mathcal{A}rgs'^+$. Then from Proposition 3 it follows that $\mathcal{A}rgs'$ is a stable extension of $AF|_{\mathcal{A}rgs' \cup \mathcal{A}rgs'^+}$. But this means that $AF|_{\mathcal{A}rgs \cup \mathcal{A}rgs^+}$ is not a *maximal* subframework that has at least one stable extension. Contradiction. $\square$

In order to understand the difference between stage semantics and the admissibility based semantics, it is useful to make an analogy with classical logic. In the presence of a potentially inconsistent knowledge base one could do two things:

1. Take the maximal consistent subsets of the knowledge base, and examine what is entailed by all of these (the “maximal consistency approach”). That is, take the (classical) models of the maximal subsets of the knowledge base that have classical models.
2. Define a new semantics such that the entire knowledge base will have models (the “new semantics approach”). This is the approach that is, for instance, taken in the field of paraconsistent logic [1,9].

Solution 1 (applying the original semantics to maximal subsets of the original problem description) is comparable to stage semantics, whereas solution 2 (redefining the semantics so that it can meaningfully be applied to a bigger class of knowledge bases) is comparable with the admissibility based semantics.

To understand the difference between solution 1 (the maximal consistency approach) and solution 2 (the new semantics approach), it helps to study the following labelling-based definition of stable semantics.

Definition 9. *Let $AF = (Ar, att)$ be an argumentation framework. A* stable labelling *is a function that assigns each argument $A \in Ar$ either the label* `in` *or* `out`*, such that:*

1. *A is labelled* `in` *iff all its attackers are labelled* `out`*, and*
2. *A is labelled* `out` *iff it has at least one attacker that is labelled* `in`*.*

The innovation of complete semantics[5] can be described as adding a third kind of label (`undec`) to the existing labels (`in` and `out`), while keeping the two clauses in the above definition the same [4,8]. A similar approach has been stated in the field of logic programming, where complete semantics is known as the *three-valued stable model semantics* [18]. In either case, the result is that under the new semantics (complete or 3-valued stable) solutions (models or extensions) exist, even for situations where solutions did not exist under the old semantics (2-valued stable). A similar trend can be observed in the field of paraconsistent logic, where some approaches try to warrant the existence of solutions (models) by implementing additional truth values [1,9].

An alternative approach would be not to come up with a fundamentally new semantics, but instead to apply the “traditional” semantics on the maximal part of the knowledge base that has “traditional” solutions. In the domain of classical logic, for instance, one could examine what is entailed by all maximal consistent subsets of formulas in the knowledge base, which in essence is the same as considering the classical models of all maximal subsets of formulas that have classical models. Similarly, in the context of logic programming, one could apply stable model semantics to the maximal subsets of a logic program that have stable models, or in the context of abstract argumentation, one could apply stable semantics to maximal subframeworks that have stable extensions, as is implemented by stage semantics.

5. Discussion

In the current paper, we have re-examined the concept of stage semantics and studied some of its properties. Apart from that, we have provided an algorithm that computes

[5]Recall that other admissibility-based semantics (like grounded, preferred, ideal or semi-stable) in essence select particular subsets of the complete extensions (labelings).

all stage labellings, and therefore also all stage extensions. This algorithm starts with the all-`in` labelling and then performs a sequence of transition steps in which the set of `in`-labelled arguments decreases and the set of `undec`-labelled arguments increases. This approach allows one to prune the search space, a possibility that would not be available if one would, for instance, start with the all-`undec` labelling, and then perform an alternative type of transition steps which increase the sets of `in` and `out`-labelled arguments. Nor would pruning be available when one uses the extensions approach (instead of the labellings approach) starting with the empty set while subsequently adding arguments such that the set remains conflict-free.

Although for the extensions approach it would also be possible to allow for pruning by starting with the set of all arguments, and then subsequently removing arguments until the set becomes conflict-free, such an approach would require the computation of the range of the set after every removal. This computation is relatively expensive, because it is essentially a global recomputation from scratch. With the labellings approach, on the other hand, no such global recomputation is needed. Instead of removing an argument (say A) from the set, we perform a transition step on an `in`-labelled argument. We do this by relabelling the argument from `in` to `out` (which is similar to removing the argument from the set) and by subsequently relabelling the `out`-labelled arguments in $\{A\} \cup A^+$ that have now become illegally `out` to `undec` (which serves the same purpose as recalculating the range of the new set). However, while recalculating the range of the new set is a global operation, based on the entire argumentation framework, relabelling any illegally `out`-labelled arguments in $\{A\} \cup A^+$ to `undec` only requires a local operation on a restricted part of the argumentation framework. This is one of the main advantages of the labellings approach above the sets approach.

Like was mentioned before, stage semantics forms one of the foundations of Verheij's DEFLOG system [16,17], which can be seen as a generalisation of Dung's notion of an argumentation framework by providing a full logical formalism of justification and attack. Where various recent work in abstract argumentation theory has been driven to implement things like higher order attacks [2] and extended argumentation frameworks [12,13], these concepts have been implemented in DEFLOG already ten years ago, in the context of stage semantics, as well as in the context of other semantics. Apart from this, the concept of semi-stable semantics can be traced back to [15], where it was described in terms of *admissible stage extensions*. Although differences in basic formalisation do not make it immediately obvious (Verheij for instance does not use the standard extensions approach) it can be proved that Verheij's approach is equivalent to that of Caminada, who independently from Verheij rediscovered the same concept, this time under the name of semi-stable semantics [5]. In addition, Caminada has proved various additional properties (like the fact that each semi-stable extension is also a complete extension [5]) and provided an algorithm [6].

One of the more fundamental issues that were treated in this paper is the difference between the "maximal consistency" approach and the "new semantics" approach. For instance, scholars in the field of paraconsistent logic had to justify their often more elaborate new semantical approaches above the simpler approach of selecting the maximal consistent subsets of a knowledge base. However, in argumentation it appears that we have gone directly to the "new semantics" approach (admissibility) without even considering the "maximal consistency" approach (stage semantics) in any serious way. This is remarkable, especially since the maximal consistency approach turns out to be expressible using the relatively simple notion of a conflict-free set with maximal range, which does not require concepts like acceptability, fixpoints and monotonic functions. This raises the question of what are the fundamental advantages of the admissibility based semantics above what appears to be the simpler approach of stage semantics.

References

[1] O. Arieli and A. Avron. The value of the four values. *Artificial Intelligence*, 102:97–141, 1998.

[2] P. Baroni, F. Cerutti, M. Giacomin, and G. Guida. Encompassing attacks to attacks in abstract argumentation frameworks. In *Proc. of ECSQARU 2009, 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 83–94, 2009.

[3] P. Baroni, M. Giacomin, and G. Guida. SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence*, 168(1-2):165–210, 2005.

[4] M.W.A. Caminada. On the issue of reinstatement in argumentation. In M. Fischer, W. van der Hoek, B. Konev, and A. Lisitsa, editors, *Logics in Artificial Intelligence; 10th European Conference, JELIA 2006*, pages 111–123. Springer, 2006. LNAI 4160.

[5] M.W.A. Caminada. Semi-stable semantics. In P.E. Dunne and TJ.M. Bench-Capon, editors, *Computational Models of Argument; Proceedings of COMMA 2006*, pages 121–130. IOS Press, 2006.

[6] M.W.A. Caminada. An algorithm for computing semi-stable semantics. In *Proceedings of the 9th European Conference on Symbolic and Quantitalive Approaches to Reasoning with Uncertainty (ECSQARU 2007)*, number 4724 in Springer Lecture Notes in AI, pages 222–234, Berlin, 2007. Springer Verlag.

[7] M.W.A. Caminada. A labelling approach for ideal and stage semantics. submitted, 2010.

[8] M.W.A. Caminada and D.M. Gabbay. A logical account of formal argumentation. *Studia Logica*, 93(2-3):109–145, 2009. Special issue: new ideas in argumentation theory.

[9] W. Carnielli, M.E. Coniglio, and J. Marcos. Logics of formal inconsistency. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, second edition*, volume 14, pages 15–114. Springer Verlag, 2002.

[10] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[11] P. M. Dung, P. Mancarella, and F. Toni. Computing ideal sceptical argumentation. *Artificial Intelligence*, 171(10-15):642–674, 2007.

[12] S. Modgil. An abstract theory of argumentation that accommodates defeasible reasoning about preferences. In *Proc. ECSQARU 2007*, pages 648–659, 2007.

[13] S. Modgil. Reasoning about preferences in argumentation frameworks. *Artificial Intelligence*, 173:901–1040, 2009.

[14] S. Modgil and M.W.A. Caminada. Proof theories and algorithms for abstract argumentation frameworks. In I. Rahwan and G.R. Simari, editors, *Argumentation in Artificial Intelligence*, pages 105–129. Springer, 2009.

[15] B. Verheij. Two approaches to dialectical argumentation: admissible sets and argumentation stages. In J.-J.Ch. Meyer and L.C. van der Gaag, editors, *Proceedings of the Eighth Dutch Conference on Artificial Intelligence (NAIC'96)*, pages 357–368, Utrecht, 1996. Utrecht University.

[16] B. Verheij. DEFLOG - a logic of dialectical justification and defeat. Technical report, Department of Metajuridica, Universiteit Maastricht, 2000.

[17] B. Verheij. DEFLOG: on the logical interpretation of prima facie justified assumptions. *Journal of Logic and Computation*, 13:319–346, 2003.

[18] Y. Wu, M.W.A. Caminada, and D.M. Gabbay. Complete extensions in argumentation coincide with 3-valued stable models in logic programming. *Studia Logica*, 93(1-2):383–403, 2009. Special issue: new ideas in argumentation theory.