Appendix A Guidelines for Stepping

In what follows, we give advice on how users can exploit stepping for analysing and debugging their code. Fig. A 1 synthesises practical guidelines for stepping from the methodological aspects of stepping described in Section 4.6. It can be seen as a user-oriented view on the stepping technique. Depending on the goals and the knowledge of the user, this guide gives concise yet high-level suggestions on how to proceed in a stepping session. The upper area of the figure is concerned with clarifying the best strategy for a stepping session and for choosing the computation to start from. The lower area, on the other hand, guides the user through the stepping process.

The diagram differentiates between four tasks a user may want to perform.

(i) Debugging a program lacking a particular answer set: we suggest to step and jump through rules that one thinks build up this answer set. Eventually, the computation will get stuck when adding a rule that prevents the answer set.

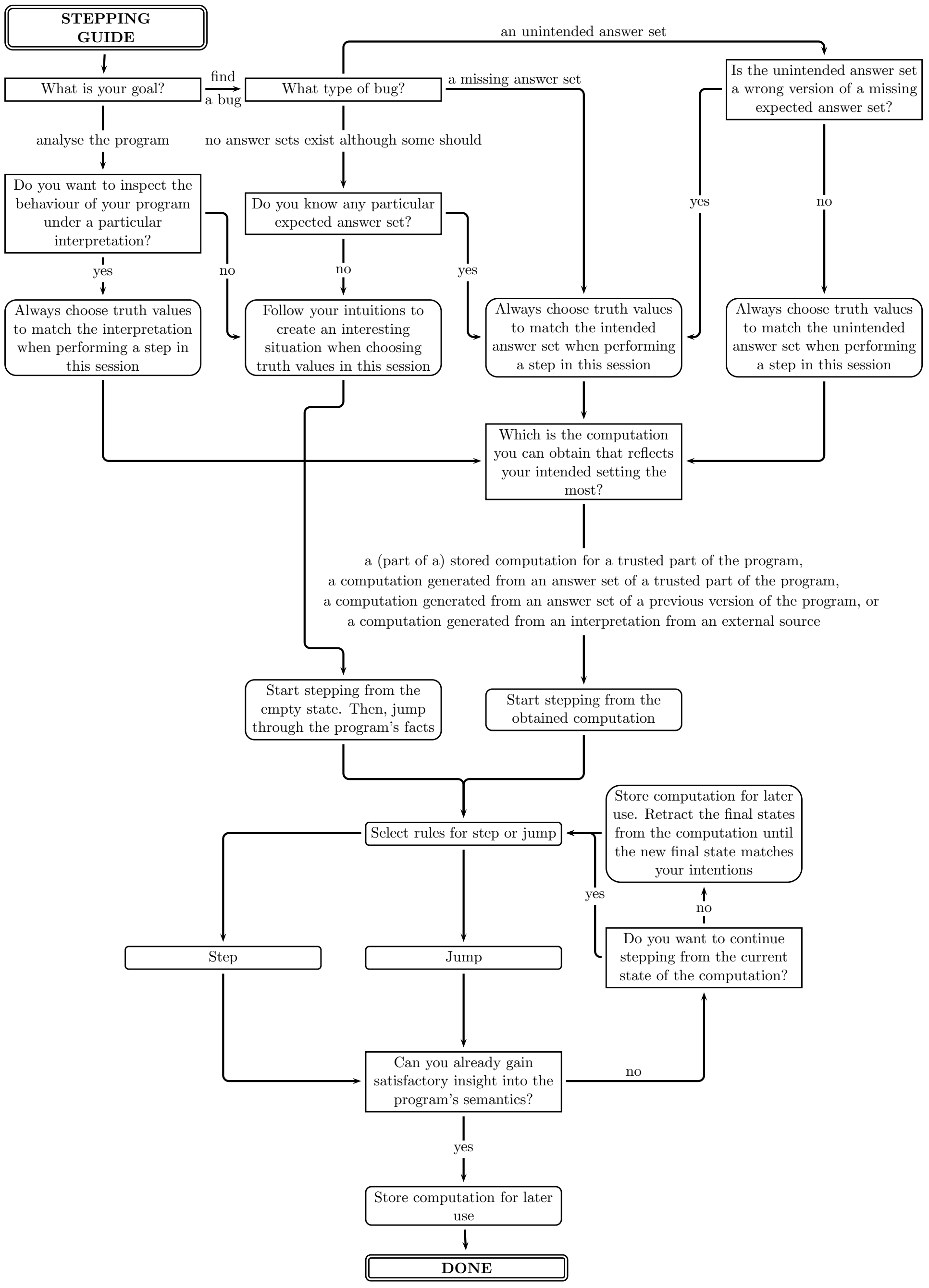
(ii) Debugging a program that lacks any answer set: if an intended answer set is known, we advise using the strategy of Item (i). Otherwise, the user should choose rules and truth values during stepping that he or she thinks should be consistent, i.e., lead to a successful computation. Also here, the computation is guaranteed to fail and get stuck, indicating a reason for the inconsistency.

(iii) Debugging a program with an unintended answer set I: In case that I is similar to an intended but missing answer set I', thus if I is intuitively a wrong version of I', we recommend stepping towards I', following the strategy of Item (i). Otherwise, the user can step towards I. Unlike in the previous cases, the computation is guaranteed to eventually succeed. Here, stepping acts as a disciplined way to inspect how the atoms of I can be derived and why no rule prevents I from being an answer set. If I is intended to be a model but not stable, then the stepping process will reveal which rules provide external support for sets of atoms that are supposed to be unfounded.

(iv) Analysing a program: In case that the user is interested in the behaviour of the program under a particular interpretation, it is reasonable to step towards this interpretation. Otherwise, rules and truth assignments should be chosen that drive the computation towards states that the user is interested in.

The procedures suggested above and in Fig. A 1 are meant as rough guidelines for the inexperienced user. Presumably, knowledge about the own source code and some practice in stepping gives the user a good intuition on how to find bugs efficiently.

It is natural to ask how big a program can get such that it is still suitable for stepping. Due to the vague nature of the question, answers cannot be clearly established. From a complexity theoretic point of view, the problems that need to be solved in a stepping support environment for and after performing a step or a jump, e.g., computing a new state from a jump, determining rules with active instances, or checking whether a computation has failed, are not harder than computing an answer set of the program under development. Under this observation, our technique is certainly an appropriate approach for debugging ASP. In some applications, however, solving times of multiple minutes or even hours are acceptable. Certainly,



STEPPING GUIDE
What is your goal?
find a bug
What type of bug?
an unintended answer set
a missing answer set
Is the unintended answer set a wrong version of a missing expected answer set?
analyse the program
no answer sets exist although some should
Do you want to inspect the behaviour of your program under a particular interpretation?
Do you know any particular expected answer set?
yes
no
no
yes
yes
no
Always choose truth values to match the interpretation when performing a step in this session
Follow your intuitions to create an interesting situation when choosing truth values in this session
Always choose truth values to match the intended answer set when performing a step in this session
Always choose truth values to match the unintended answer set when performing a step in this session
Which is the computation you can obtain that reflects your intended setting the most?
a (part of a) stored computation for a trusted part of the program,
a computation generated from an answer set of a trusted part of the program,
a computation generated from an answer set of a previous version of the program, or
a computation generated from an interpretation from an external source
Start stepping from the empty state. Then, jump through the program's facts
Start stepping from the obtained computation
Select rules for step or jump
Store computation for later use. Retract the final states from the computation until the new final state matches your intentions
yes
no
Step
Jump
Do you want to continue stepping from the current state of the computation?
Can you already gain satisfactory insight into the program's semantics?
no
yes
Store computation for later use
DONE

Fig. A 1. Stepping guide

having waiting times of these lengths for individual debugging steps is undesirable. On the positive side, often, following a few guidelines during the development of an answer-set program can significantly reduce the likelihood of introducing bugs, the amount of information the user has to deal with, and also the computational resources required for stepping. Among these measures are best practices for ASP development that have been discussed in a paper by Brain et al. (2009). For working with the stepping method in particular, we give the following recommendations.

Use scalable encodings and start with small examples. Using small problem instances, also the resulting grounding as well as answer sets are typically small. This limits the amount of information to be considered during debugging. Chances that bugs are detected early, using small programs is suggested by an evaluation of the *small-scope hypothesis* for ASP (Oetsch et al. 2012).

Visualise answer sets and stepping states. Tools like `Kara` (Kloimüllner et al. 2013) (that is implemented in `SeaLion`), `ASPVIZ` (Cliffe et al. 2008), `IDPDraw` (Wittocx 2009), or `Lonsdaleite` (Smith 2011) allow for visualising interpretations. With their help, one can quickly spot when an answer set differs from what is expected and they allow to monitor the evolvement of the interpretation that is build up during stepping. The illustrations of the maze generation problem in this section were created using `Kara`. For use with stepping, we advise to specify visualisations also for interpretations that are not supposed to be answer sets. For example, in Fig. 8, we have visualisations for cells that are not assigned to be empty or a wall and for cells that are assigned to be a wall and empty, despite in an expected answer set, every cell has to be either a wall or empty.

Test often. Frequent tests allow the user to trust in large parts of the program, hence these parts can be jumped over in a stepping session.

Appendix B Remaining Proofs

Theorem 1
Let S be a state and S' a successor of S, where $\Delta = I_{S'} \setminus I_S$. Moreover, let X' be a set of literals with $\emptyset \subset X' \subseteq I_{S'}$. Then, the following statements are equivalent:

(i) X' is unfounded in $P_{S'}$ with respect to $I_{S'}$.
(ii) $X' = \Delta' \cup X$, where $\Delta' \subseteq \Delta$, $X \in \Upsilon_S$, and $r_{new}(S, S')$ is not an external support for X' with respect to $I_{S'}$.

Proof
((i)$\Rightarrow$(ii)) It is obvious that $r_{new}(S, S')$ is not an external support for X' with respect to $I_{S'}$ as otherwise X' cannot be unfounded in $P_{S'}$ with respect to $I_{S'}$. It remains to be shown that $X' = \Delta' \cup X$ for some $\Delta' \subseteq \Delta$ and some $X \in \Upsilon_S$. Towards a contradiction, assume $X' \neq \Delta'' \cup X''$ for all $X'' \in \Upsilon_S$ and all $\Delta'' \subseteq \Delta$. We define $X = X' \cap I_S$.

Consider the case that $X \in \Upsilon_S$. As $X' \setminus I_S \subseteq \Delta$, and $X' = (X' \setminus I_S) \cup X$, we have a contradiction to our assumption. Therefore, it holds that $X \notin \Upsilon_S$. Hence, as $X \subseteq I_S$, by definition of a state, X is not unfounded in P_S with respect to I_S. Therefore, there is some external support $r \in P_S$ for X with respect to I_S.

In the following, we show that r is also an external support for X' with respect to $I_{S'}$. Since S' is a successor of S and S is a state, we get that I_S and $I_{S'}$ coincide on D_r. Consequently, from $I_S \models \mathrm{B}(r)$ we get that also $I_{S'} \models \mathrm{B}(r)$. Moreover, because of $I_S \setminus X \models \mathrm{B}(r)$ it is also true that $I_{S'} \setminus X' \models \mathrm{B}(r)$. Furthermore, we know that there is some $A \in \mathrm{H}(r)$ with $X|_{D_A} \neq \emptyset$ and $I_S|_{D_A} \subseteq C$, for some $C \in C_A$. As $X|_{D_A} = X'|_{D_A}$ and $I_S|_{D_A} = I_{S'}|_{D_A}$ we also have $X'|_{D_A} \neq \emptyset$ and $I_{S'}|_{D_A} \subseteq C$. Finally, note that for all $A \in \mathrm{H}(r)$ with $I_S \models A$, we have $(X \cap I_S)|_{D_A} \neq \emptyset$. Consider some $A \in \mathrm{H}(r)$ such that $I_{S'} \models A$. From the latter we get that $I_S \models A$ and therefore $(X \cap I_S)|_{D_A} \neq \emptyset$. As $X \cap I_S \subseteq X' \cap I_{S'}$, we also have $(X' \cap I_{S'})|_{D_A} \neq \emptyset$. Hence, r fulfils all conditions for being an external support for X' with respect to $I_{S'}$, which is a contradiction to X' being unfounded in $P_{S'}$ with respect to $I_{S'}$.

((ii)$\Rightarrow$(i)) Towards a contradiction, assume X' has some external support $r \in P_{S'}$ with respect to $I_{S'}$. From (ii) we know that $r \neq r_{new}(S, S')$ and $X' = \Delta' \cup X$ for some $\Delta' \subseteq \Delta$ and some $X \in \Upsilon_S$. As $r \neq r_{new}(S, S')$, we have that I_S and $I_{S'}$ coincide on D_r. Therefore, from $I_{S'} \models \mathrm{B}(r)$ and $I_{S'} \setminus X' \models \mathrm{B}(r)$, it follows that $I_S \models \mathrm{B}(r)$ and $I_S \setminus X' \models \mathrm{B}(r)$. Note that $X = X' \cap I_S$ and hence $I_S \setminus X \models \mathrm{B}(r)$. We know that there is some $A \in \mathrm{H}(r)$ with $X'|_{D_A} \neq \emptyset$ and $I_{S'}|_{D_A} \subseteq C$, for some $C \in C_A$. As $X'|_{D_A} = X|_{D_A}$ we have $X|_{D_A} \neq \emptyset$. Moreover, as $I_{S'}|_{D_A} = I_S|_{D_A}$, it holds that $I_S|_{D_A} \subseteq C$. Finally, notice that for all $A \in \mathrm{H}(r)$ with $I_{S'} \models A$, we have $(X' \cap I_{S'})|_{D_A} \neq \emptyset$. Consider some $A \in \mathrm{H}(r)$ with $I_S \models A$. As $I_{S'}|_{D_A} = I_S|_{D_A}$, we also have $I_{S'} \models A$ and hence $(X' \cap I_{S'})|_{D_A} \neq \emptyset$. As $D_A \cap \Delta = \emptyset$, we have $(X' \cap I_{S'})|_{D_A} = (X \cap I_S)|_{D_A}$. Consequently, it holds that $(X \cap I_S)|_{D_A} \neq \emptyset$. We showed that r is an external support of X in P_S with respect to I_S. Therefore, we have a contradiction to $X \in \Upsilon_S$ because S is a state. $\square$

Theorem 3

Let S_0 be a state, P a C-program with $P_{S_0} \subseteq P$, and I an answer set of P with $I_{S_0} \subseteq I$ and $I \cap I^-{}_{S_0} = \emptyset$. Then, there is a computation $S_0, \ldots, S_n$ that has succeeded for P such that $P_{S_n} = P^I$ and $I_{S_n} = I$.

Proof

The proof is by induction on the size of the set $P^I \setminus P_{S_0}$. Observe that from $I_{S_0} \subseteq I$, $I \cap I^-{}_{S_0} = \emptyset$, and $I_{S_0} \models \mathrm{B}(r)$ and $D_r \subseteq I_{S_0} \cup I^-{}_{S_0}$, for all $r \in P_{S_0}$, we get that $I \models \mathrm{B}(r)$ for all $r \in P_{S_0}$. Hence, as $P_{S_0} \subseteq P$, we have $P_{S_0} \subseteq P^I$.

Consider the base case that $|P^I \setminus P_{S_0}| = 0$. From $P_{S_0} \subseteq P^I$ we get $P_{S_0} = P^I$. Consider the sequence $C = \langle P_{S_0}, I_{S_0}, I^-{}_{S_0}, \Upsilon_{S_0} \rangle$. Towards a contradiction, assume $I_{S_0} \neq I$. As $I_{S_0} \subseteq I$ this means $I_{S_0} \subset I$. Hence, there is some $a \in I \setminus I_{S_0}$. As for all $r \in P_{S_0}$ it holds that $D_r \subseteq I_{S_0} \cup I^-{}_{S_0}$, and $I \cap I^-{}_{S_0} = \emptyset$, we get $a \notin D_{P_{S_0}}$. We have a contradiction to $I \in AS(P_{S_0})$ by Corollary 1, as $\{a\}$ is unfounded in P_{S_0} with respect to I. Consequently, $I_{S_0} = I$ must hold. As I_{S_0} is an answer set of P_{S_0}

and S_0 is a state, we have that $\Upsilon_{S_0} = \{\emptyset\}$ by definition of state. It follows that C meets the criteria of the conjectured computation.

We proceed with the step case. As induction hypothesis, assume that the claim holds whenever $|P^I \setminus P_{S_0}| \leq i$ for an arbitrary but fixed $i \geq 0$. Consider some state S_0 and some $I \in AS(P_{S_0})$ for which the conditions in the premise hold such that $|P^I \setminus P_{S_0}| = i + 1$. Towards a contradiction, assume there is no C-rule $r \in P^I \setminus P_{S_0}$ such that $I_{S_0} \models \mathrm{B}(r)$. Note that there is at least one C-rule $r' \in P^I \setminus P_{S_0}$ because $|P^I \setminus P_{S_0}| = i + 1$. It cannot hold that $I = I_{S_0}$ since from $r' \in P^{I_{S_0}}$ follows $I_{S_0} \models \mathrm{B}(r')$. Consequently, we have $I_{S_0} \subset I$. Consider some $r'' \in P^I$ with $I_{S_0} \models \mathrm{B}(r'')$. By our assumption, we get that $r'' \in P_{S_0}$. It follows that $I_{S_0} \models r''$, and consequently there is some C-atom $A \in \mathrm{H}(r'')$ with $I_{S_0} \models A$. As $D_{r''} \subseteq D_{S_0}$, we have $D_A \subseteq I_{S_0} \cup {I^-}_{S_0}$. From that, since $I_{S_0} \subset I$ and $I \cap {I^-}_{S_0} = \emptyset$, we get $I|_{D_A} = I_{S_0}|_{D_A}$. We have a contradiction to I being an answer set of P by Definition 12.

So, there must be some C-rule $r \in P^I \setminus P_{S_0}$ such that $I_{S_0} \models \mathrm{B}(r)$. From $r \in P^I$ we get $I \models \mathrm{B}(r)$ and $I \models r$. Consider the state structure $S_1 = \langle P_1, I_1, {I_1}^-, \Upsilon_1 \rangle$, where $P_1 = P_{S_0} \cup \{r\}$, $I_1 = I_{S_0} \cup (I \cap D_r)$, ${I_1}^- = {I^-}_{S_0} \cup (D_r \setminus I)$, and

$$\begin{aligned}\Upsilon_1 = \{X \mid & X = \Delta' \cup X', \text{ where } \Delta' \subseteq (I_1 \setminus I_{S_0}), X' \in \Upsilon_{S_0}, \text{ and} \\ & r \text{ is not an external support of } X \text{ with respect to } I_1\}\cdot\end{aligned}$$

S_1 is a successor of state S_0, therefore S_1 is also a state by Corollary 2. As $P_1 \subseteq P$, $I_1 \subseteq I$, $I \cap {I_1}^- = \emptyset$, and $|P^I \setminus P_1| = i$, by the induction hypothesis, $S_1, \ldots, S_n$ is a computation, where S_n is a stable state, $P_{S_n} = P^I$, and $I_{S_n} = I$. Since S_1 is a successor of state S_0, also $S_0, S_1, \ldots, S_n$ is a computation. $\square$

For establishing Theorem 5 we make use of the following notion which reflects positive dependency on the rule level.

Definition 1
The *positive rule dependency graph* of P is given by

$$G_{\mathfrak{R}}(P) = \langle P, \{\langle r_1, r_2 \rangle \mid r_1, r_2 \in P, posOcc(\mathrm{B}(r_1)) \cap posOcc(\mathrm{H}(r_2)) \neq \emptyset\}\rangle\cdot$$

We can relate the two notions of dependency graph as follows.

Lemma 1
Let P be a C-program. $G_{\mathfrak{R}}(P)$ is acyclic iff $G(P)$ is acyclic.

Proof
Let $\prec_{\mathfrak{D}}$ denote the edge relation of $G(P)$ and $\prec_{\mathfrak{R}}$ that of $G_{\mathfrak{R}}(P)$.

($\Rightarrow$) Assume $G(P)$ is not acyclic. There must be some path $a_1, \ldots, a_n$ of atoms a_i such that for $1 \leq i < n$, we have $a_i \in D_P$, $a_i \prec_{\mathfrak{D}} a_{i+1}$, and $a_1 = a_n$. Hence, by the definition of $G(P)$, there must be a sequence $r_1, \ldots, r_{n-1}$ such that for each $1 \leq i \leq n-1$, $r_i \in P$, $a_i \in posOcc(\mathrm{H}(r_i))$, and $a_{i+1} \in posOcc(\mathrm{B}(r_i))$. Therefore, for each $1 \leq i < n-1$, we have $r_{i+1} \prec_{\mathfrak{R}} r_i$. Note that $a_1 \in posOcc(\mathrm{H}(r_1))$ and $a_1 \in posOcc(\mathrm{B}(r_{n-1}))$. Consequently, we have $r_{n-1} \prec_{\mathfrak{R}} r_1$ and thus $r_1, r_{n-1}, \ldots, r_1$ forms a cycle in $G_{\mathfrak{R}}(P)$. It follows that $G_{\mathfrak{R}}(P)$ is not acyclic.

($\Leftarrow$) Assume now that $G_{\mathfrak{R}}(P)$ is not acyclic. There must be some path $r_1, \ldots, r_n$ of C-rules r_i such that for $1 \leq i < n$ we have $r_i \in P$, $r_1 = r_n$, and $r_i \prec_{\mathfrak{R}} r_{i+1}$.

Hence, by the definition of $G_{\mathfrak{R}}(P)$, there must be a sequence $a_1, \ldots, a_{n-1}$ such that for each $1 \leq i \leq n-1$, $a_i \in \mathit{posOcc}(\mathrm{H}(r_{i+1}))$, and $a_i \in \mathit{posOcc}(\mathrm{B}(r_i))$. Therefore, for each $1 \leq i < n-1$ we have $a_{i+1} \prec_{\mathfrak{D}} a_i$. Note that $a_{n-1} \in \mathit{posOcc}(\mathrm{H}(r_1))$ and $a_1 \in \mathit{posOcc}(\mathrm{B}(r_1))$. Consequently, we have $a_{n-1} \prec_{\mathfrak{D}} a_1$ and thus $a_1, a_{n-1}, \ldots, a_1$ forms a cycle in $G(P)$. We have that $G_{\mathfrak{R}}(P)$ is not acyclic. $\square$

Lemma 2
Let P be an absolutely tight C-program. There is a strict total order $\prec$ on P that extends the reachability relation of $G_{\mathfrak{R}}(P)$.

Proof
By Definition 21, $G(P)$ is acyclic. Hence, by Lemma 1, $G_{\mathfrak{R}}(P)$ is also acyclic. The conjecture holds, since every directed acyclic tree has a topological ordering. $\square$

We now have the means to show Theorem 5, guaranteeing the existence of stable computations.

Theorem 5
Let $\mathrm{C} = S_0, \ldots, S_n$ be a computation such that S_0 and S_n are stable and $P_\Delta = P_{S_n} \setminus P_{S_0}$ is a normal, convex, and absolutely tight C-program. Then, there is a stable computation $\mathrm{C}' = S'_0, \ldots, S'_n$ such that $S_0 = S'_0$ and $S_n = S'_n$.

Proof
Let $\prec$ be the strict total order extending the reachability relation of $G_{\mathfrak{R}}(P_\Delta)$ that is guaranteed to exist by Lemma 2. Let $r(\cdot) : \{1, \ldots, n\} \mapsto P_\Delta$ denote the one-to-one mapping from the integer interval $\{1, \ldots, n\}$ to the C-rules from P_Δ such that for all i, j in the range of $r(\cdot)$, we have that $i < j$ implies $r(j) \prec r(i)$. Consider the sequence $\mathrm{C}' = S'_0, \ldots, S'_n$, where $S'_0 = S_0$, and for all $0 \leq i < n$,

$$\begin{aligned}
P'_{i+1} &= P'_i \cup \{r(i+1)\},\\
I_{S'_{i+1}} &= I_{S'_i} \cup (I_{S_n} \cap D_{r(i+1)}),\\
I^-{}_{S'_{i+1}} &= I^-{}_{S'_i} \cup (I^-{}_{S_n} \cap D_{r(i+1)}), \quad \text{and}\\
\Upsilon_{S'_{i+1}} &= \{\emptyset\}.
\end{aligned}$$

Notice that $S'_n = S_n$ and $I_{S'_{i+1}}|_{D_{P_{S'_i}}} = I_{S'_i}|_{D_{P_{S'_i}}}$, for all $0 \leq i < n$. We show that C' is a computation by induction on the length of a subsequence of C'.

As base case consider the sequence $\mathrm{C}'' = S'_0$. As $S'_0 = S_0$ and S_0 is a state, C'' is a computation. For the induction hypothesis, assume that for some arbitrary but fixed i with $0 \leq i < n$, the sequence $S'_0, \ldots, S'_i$ is a computation.

In the induction step it remains to be shown that S'_{i+1} is a successor of S'_i. Clearly, S'_{i+1} is a state structure, and by definition of C', since C is a computation and

$$I_{S'_{i+1}}|_{D_{P_{S_{i+1}}}} = I_{S_n}|_{D_{P_{S_{i+1}}}},$$

Conditions (i), (ii), (iii), and (v) of Definition 17 for being a successor of S'_i are fulfilled by S'_{i+1}. Let Δ denote $I_{S'_{i+1}} \setminus I_{S'_i}$.

Next we show that Condition (iv) holds, i.e., $I_{S'_i} \models \mathrm{B}(r(i+1))$. Note that since Condition (v) holds, we have $I_{S'_{i+1}} \models \mathrm{B}(r(i+1))$ and hence (iv) holds in the case $\Delta = \emptyset$. Towards a contradiction assume $\Delta \neq \emptyset$ and $I_{S'_i} \not\models \mathrm{B}(r(i+1))$. We define $\Delta_{B^+} = \Delta \cap \mathit{posOcc}(\mathrm{B}(r(i+1)))$.

First, consider the case that $\Delta_{B^+} = \emptyset$. As $I_{S'_i} \not\models \mathrm{B}(r(i+1))$, there must be some C-literal $L \in \mathrm{B}(r(i+1))$ such that $I_{S'_i} \not\models L$. We know that $I_{S'_{i+1}} \models L$. Consequently, $I_{S'_i}|_{D_L} \subset I_{S'_{i+1}}|_{D_L}$ and therefore $\Delta|_{D_L} \neq \emptyset$. Moreover, from $I_{S'_{i+1}} \models L$ we have

$$I_{S'_{i+1}}|_{D_L} \subseteq \mathit{posOcc}(\mathrm{B}(r(i+1)))\cdot$$

It follows that $\Delta|_{D_L} \cap \mathit{posOcc}(\mathrm{B}(r(i+1))) \neq \emptyset$, indicating a contradiction to $\Delta_{B^+} = \emptyset$. It holds that $\Delta_{B^+} \neq \emptyset$. Note that $X \subseteq I_{S_n}$. From that, since S_n is a state, there must be some C-rule $r_{\Delta_{B^+}} \in P_{S_n}$ such that $r_{\Delta_{B^+}}$ is an external support for Δ_{B^+} with respect to I_{S_n}. It cannot be the case that $r \in P_{S_0}$, since $\Delta_{B^+} \cap I_{S'_i} = \emptyset$, therefore, $r_{\Delta_{B^+}} \in P_\Delta$. As $r_{\Delta_{B^+}}$ is an external support for Δ_{B^+} with respect to I_{S_n}, for $\{A\} = \mathrm{H}(r_{\Delta_{B^+}})$, we have $I_{S_n} \models A$ and $\Delta_{B^+}|_{D_A} \neq \emptyset$.

Consider the case that $r_{\Delta_{B^+}} = r(i+1)$. From that we get $\mathit{posOcc}(\mathrm{H}(r(i+1))) \cap \Delta_{B^+} \neq \emptyset$. This, in turn, implies $\mathit{posOcc}(\mathrm{H}(r(i+1))) \cap \mathit{posOcc}(\mathrm{B}(r(i+1))) \neq \emptyset$ which is a contradiction to $G_{\mathfrak{R}}(P_\Delta)$ being acyclic. The latter is guaranteed by absolute tightness of P_Δ and Lemma 1.

Consider the case that $r(i+1) \prec r_{\Delta_{B^+}}$. Then, by definition of C' we have that $r_{\Delta_{B^+}} \in P_{S'_i}$. Hence, from $\Delta_{B^+}|_{D_A} \neq \emptyset$ follows

$$\Delta_{B^+}|_{D_{P_{S'_i}}} \neq \emptyset \qquad \text{and thus} \qquad I_{S'_{i+1}} \setminus I_{S'_i}|_{D_{P_{S'_i}}} \neq \emptyset\cdot$$

The latter is a contradiction to $I_{S'_{i+1}}|_{D_{P_{S'_i}}} = I_{S'_i}|_{D_{P_{S'_i}}}$.

Consider the remaining case that $r_{\Delta_{B^+}} \prec r(i+1)$. As $\Delta_{B^+}|_{D_A} \neq \emptyset$, $\Delta_{B^+} \subseteq I_{S_n}$, and $I_{S_n}|_{D_A} \in C_A$, it holds that $\mathit{posOcc}(\mathrm{H}(r_{\Delta_{B^+}})) \cap \Delta_{B^+} \neq \emptyset$. Therefore, we have $\mathit{posOcc}(\mathrm{H}(r_{\Delta_{B^+}})) \cap \mathit{posOcc}(\mathrm{B}(r(i+1))) \neq \emptyset$. This implies $r(i+1) \prec r_{\Delta_{B^+}}$, being a contradiction to $\prec$ being a strict order as we also have $r_{\Delta_{B^+}} \prec r(i+1)$. Thus, Condition (iv) of Definition 17 for being a successor of S'_i holds for S'_{i+1}.

Towards a contradiction assume Condition (vi) does not hold. Hence, it must hold that there is some $\Delta' \subseteq \Delta$ such that $\Delta' \neq \emptyset$ and $r(i+1)$ is not an external support for Δ' with respect to $I_{S'_{i+1}}$. We have $I_{S'_{i+1}} \models \mathrm{B}(r(i+1))$ and since we already know that $I_{S'_i} \models \mathrm{B}(r(i+1))$, also $I_{S'_{i+1}} \setminus \Delta' \models \mathrm{B}(r(i+1))$ holds by convexity of P_Δ. Moreover, as $I_{S'_{i+1}} \models r(i+1)$, it must hold that $I_{S'_{i+1}} \models A$ for $\mathrm{H}(r(i+1)) = \{A\}$. Consequently, for $r(i+1)$ not to be an external support for Δ' with respect to $I_{S'_{i+1}}$, we have $\Delta'|_{D_A} = \emptyset$. As then $\Delta'|_{D_{\mathrm{H}(r(i+1))}} = \emptyset$ but $\Delta'|_{D_{r(i+1)}} \neq \emptyset$ it must hold that $\Delta'|_{D_{\mathrm{B}(r(i+1))}} \neq \emptyset$. Consider $\Delta'' = \Delta' \cap \mathit{posOcc}(\mathrm{B}(r(i+1)))$ and assume that $\Delta'' \neq \emptyset$. Then, as $\Delta'' \subseteq I_{S_n}$, there must be some C-rule $r_{\Delta''}$ that is an external support for Δ'' with respect to I_{S_n}. Hence, $\mathit{posOcc}(\mathrm{H}(r_{\Delta''})) \cap \Delta'' \neq \emptyset$ and therefore $\mathit{posOcc}(\mathrm{H}(r_{\Delta''})) \cap \mathit{posOcc}(\mathrm{B}(r(i+1))) \neq \emptyset$. It follows that $r(i+1) \prec r_{\Delta''}$. From that we get $r_{\Delta''} \in P_{S'_i}$. This is a contradiction as we know that $\mathit{posOcc}(\mathrm{H}(r_{\Delta''})) \cap \Delta'' \neq \emptyset$, $\mathit{posOcc}(\mathrm{H}(r_{\Delta''})) \cap \Delta'' \subseteq I_{S'_i}$, and $\Delta'' \subseteq I_{S'_{i+1}} \setminus I_{S'_i}$. Consequently, $\Delta' \cap \mathit{posOcc}(\mathrm{B}(r(i+1))) = \emptyset$ must hold. From $\Delta'|_{D_{\mathrm{B}(r(i+1))}} \neq \emptyset$ we get that there is some $L \in \mathrm{B}(r(i+1))$ with $\Delta'|_{D_L} \neq \emptyset$. As $I_{S'_{i+1}} \models L$, we have that

$I_{S'_{i+1}}|_{D_L} \in C$ in the case L is a C-atom $L = \langle D_L, C\rangle$, and $I_{S'_{i+1}}|_{D_L} \in 2^D_L \setminus C$ in the case L is a default negated C-atom $L = \mathit{not}\ \langle D_L, C\rangle$. In both cases, as $\Delta' \subseteq I_{S'_{i+1}}$ and $\Delta'|_{D_L} \neq \emptyset$, we get a contradiction to $\Delta' \cap \mathit{posOcc}(\mathrm{B}(r(i+1))) = \emptyset$. $\square$

References

Brain, M., Cliffe, O., and De Vos, M. 2009. A pragmatic programmer's guide to answer set programming. In *Proceedings of the 2nd International Workshop on Software Engineering for Answer-Set Programming* (SEA'09), Potsdam, Germany, M. De Vos and T. Schaub, Eds. 49–63.

Cliffe, O., De Vos, M., Brain, M., and Padget, J. A. 2008. ASPVIZ: Declarative visualisation and animation using answer set programming. In *Proceedings of the 24th International Conference on Logic Programming* (ICLP'08), Udine, Italy, Dec. 9-13, 2008, M. G. de la Banda and E. Pontelli, Eds. LNCS, vol. 5366. Springer, 724–728.

Kloimüllner, C., Oetsch, J., Pührer, J., and Tompits, H. 2013. Kara: A system for visualising and visual editing of interpretations for answer-set programs. In *Revised Selected Papers of the 19th International Conference on Applications of Declarative Programming and Knowledge Management* (INAP'11) and the 25th Workshop on Logic Programming (WLP'11), Vienna, Austria, Sept. 28-30, 2011. LNCS, vol. 7773. Springer, 325–344.

Oetsch, J., Prischink, M., Pührer, J., Schwengerer, M., and Tompits, H. 2012. On the small-scope hypothesis for testing answer-set programs. In *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning* (KR'12), Rome, Italy, June 10-14, 2012, G. Brewka, T. Eiter, and S. A. McIlraith, Eds. AAAI Press.

Smith, A. 2011. Lonsdaleite. `https://github.com/rndmcnlly/Lonsdaleite`. [Online; accessed Dec. 14, 2016].

Wittocx, J. 2009. IDPDraw, a tool used for visualizing answer sets. `https://dtai.cs.kuleuven.be/software/idpdraw`. [Online; accessed Dec. 14, 2016].