Stable-Ordered Models for Propositional Theories with Order Operators

Johannes Oetsch and Juan-Carlos Nieves

Department of Computing Science,
Umeå University, Sweden
{joetsch,jcnieves}@cs.umu.se

Abstract. The stable-model semantics has been generalised from logic programs to arbitrary theories. We explore a further generalisation and consider sequences of atoms as models instead of sets. The language is extended by suitable order operators to access this additional information. We recently introduced an extension of classical logic by a single order operator with a temporal interpretation for activity reasoning. The logic envisaged here is a nonmonotonic version thereof. Our definition of what we call stable-ordered models is based on the stable-model semantics for theories due to Ferraris and Lifschitz with the necessary changes. Compared to related nonmonotonic versions of temporal logics, our approach is less costly as checking model existence remains at the second level of the polynomial hierarchy. We demonstrate versatile applications from activity reasoning, combinatorial testing, debugging concurrent programs, and digital forensics.

Keywords: stable-model semantics, temporal logic, nonmonotonic reasoning, knowledge representation

1 Introduction

Answer-set programming (ASP) is a problem solving paradigm with many AI applications [7]. It means that problems are encoded as logic programs so that the solutions correspond to the models of the programs [13, 15]. The stable-model semantics for logic programs, introduced by Gelfond and Lifschitz [10], is the basis of ASP. It has been generalised to arbitrary theories by Pearce [17] using equilibrium logic. Ferraris and Lifschitz [8, 9] came up with an alternative characterisation that is based on reducts which is more in the spirit of the original definition. We explore a further generalisation and consider sequences of atoms as models instead of sets. The language is extended by suitable order operators to access this additional information. The order of atoms can be interpreted in different ways: either temporal, e.g., the order in which goals or events are achieved, or in a more general sense as how objects of interest are arranged or permuted.

Besides curiosity, our motivation to study a new generalisation comes from *activity reasoning*, a topic that we studied in recent work [16]. There, we introduced a monotonic version of the logic we are envisioning here. Models in this monotonic logic, dubbed *ordered models*, are sequences of atoms without repetition and the language is standard classical logic plus a single modal operator. Intuitively, a model represents in which

order goals—viewed as atomic entities—are achieved. However, the monotonic semantics comes with the usual limitations for representing incomplete knowledge which is common in activity reasoning. Based on the temporal logic from previous work, we introduce a *nonmonotonic semantics for theories with order operators*. We use the reduct-based definition of stable models for theories from Ferraris and Lifschitz [8, 9] with the necessary changes to define what we call *stable-ordered models*.

Related to this work is a nonmonotonic variant of linear-temporal logic (LTL) [18] based on infinite traces and equilibrium logic [1]. A version of this logic for finite traces has been introduced recently by Cabalar et al. [5]. Their approach is readily implementable via ASP but requires multi-shot solving, i.e., several calls to an ASP solver are necessary to compute a satisfying trace. This is in accordance with the complexity of satisfiability checking which is PSPACE hard. Also other approaches extend LTL for nonmonotonic reasoning and elaboration tolerant goal representations [2, 3]. Our approach is different from all previous work as the idea to use a sequence of atoms as model is quite unique. The complexity of checking model existence remains at Σ_2^P which means a computational advantage over related work. Although our notion of stable-ordered model is less expressive than arbitrary traces of states, there are interesting applications where it suffices. We demonstrate this with examples from activity reasoning, combinatorial testing, fault detection for concurrent programs, and digital forensics.

2 Preliminaries

The logic we introduce in this paper is a nonmonotonic version of a monotonic temporal logic that we proposed in recent work in the context of activity reasoning [16]. Language $\mathcal{L}$ is determined by an infinite countable set $\mathcal{U}$ of atoms, Boolean connectives $\supset$, $\wedge$, $\vee$, $\bot$, and the modal operators $\circ$, $\Diamond$, and $\Box$, where $\circ$ means previously, $\Box$ stands for now and always in the past, and $\Diamond$ means now or at some time in the past.[1] A *formula* φ is defined by the grammar $\varphi ::= a \mid \bot \mid (\varphi_1 \supset \varphi_2) \mid (\varphi_1 \wedge \varphi_2) \mid (\varphi_1 \vee \varphi_2) \mid (\circ\varphi) \mid (\Diamond\varphi) \mid (\Box\varphi)$ where $a \in \mathcal{L}$ is an atom. Parentheses are omitted if no ambiguities arise.

A *theory* is a set of formulas. For an expression (formula or theory) e, $At(e)$ is the set of atoms occurring in e. We use $P = \langle a_1, \ldots, a_n \rangle$ to denote a finite sequence of elements. The *length* of P, denoted by $|P|$, is n. For two sequences P and Q, PQ is the concatenation of P and Q. We say that P is a *prefix* of Q and write $P \preceq Q$ iff $Q = PR$ for some sequence R. The empty sequence is denoted by ε. Note that the notion of prefix is reflexive. An *ordered model* M over $\mathcal{U}$ is a sequence of atoms from $\mathcal{U}$ without repetition. We write $M \models \varphi$ to denote that formula φ is *true* in M. Relation $\models$ is inductively defined as follows:

$M \not\models \bot$
$M \models a$ iff a occurs in M, for an atom $a \in \mathcal{U}$;
$M \models \varphi_1 \wedge \varphi_2$ iff $M \models \varphi_1$ and $M \models \varphi_2$;
$M \models \varphi_1 \vee \varphi_2$ iff $M \models \varphi_1$ or $M \models \varphi_2$;
$M \models \varphi_1 \supset \varphi_2$ iff $M \not\models \varphi_1$ or $M \models \varphi_2$;

[1] Note that $\circ$ was not part of the initial version of the logic to avoid unintended effects when used under the open-world assumption [16]. Also, note that we do not consider strong negation.

$M \models \circ\varphi$ iff $M' \models \varphi$ for some $M' \preceq M$ with $|M| - |M'| = 1$;
$M \models \Diamond\varphi$ iff $M' \models \varphi$ for some $M' \preceq M$;
$M \models \Box\varphi$ iff $M' \models \varphi$ for each $M' \preceq M$.

For a theory T, $M \models T$ iff $M \models \varphi$ for each $\varphi \in T$. Also, entailment, equivalence, etc. are defined as in classical logic. We use the abbreviations $\neg\varphi := \varphi \supset \bot$ and $\top := \neg\bot$. To denote the initial state, we define $\mathbb{I} := \neg\circ\top$ which is only true in ε.

An ordered model $\langle a_1, \ldots, a_n\rangle$ can be understood as a compact representation of a sequence $s_0, \ldots, s_n$ of classical models that represent states of the world, where $s_0 = \emptyset$ and the only difference between any s_i and s_{i+1} is that atom a_{i+1} becomes true in s_{i+1}. Note that two properties that are reasonable to assume in the context of goal achievements are implicit in this representation: First, once an atom becomes true, i.e., a goal has been achieved, it remains true (*persistence*). Second, the same goal cannot be reached twice (*Heraclitus principle*). We refer the reader to previous work for details [16].

3 Stable-Ordered Models

The nonmonotonic semantics of language $\mathcal{L}$ that we define in this paper is an adaptation of the stable-model semantics due to Ferraris and Lifschitz [8, 9] for propositional theories with the necessary changes. For a formula φ and an ordered model X over $At(\varphi)$, we define the *reduct* of φ with respect to X, φ^X in symbols, inductively as follows:

$$\bot^X = \bot$$

$$a^X = \begin{cases} a & \text{if } a \text{ is an atom and } X \models a \\ \bot & \text{otherwise} \end{cases}$$

$$(\varphi_1 \otimes \varphi_2)^X, \otimes \in \{\wedge, \vee, \supset\} = \begin{cases} \varphi_1^X \otimes \varphi_2^X & \text{if } X \models \varphi_1 \otimes \varphi_2 \\ \bot & \text{otherwise} \end{cases}$$

$$(\circ\varphi)^X = \begin{cases} \circ\varphi^Y & \text{if } Y \preceq X, |X| - |Y| = 1, \text{ and } Y \models \varphi \\ \bot & \text{otherwise} \end{cases}$$

$$(\Diamond\varphi)^X = \bigvee_{Y \in \{X' | X' \preceq X\}} \Diamond\varphi^Y$$

$$(\Box\varphi)^X = \bigwedge_{Y \in \{X' | X' \preceq X\}} \Box\varphi^Y$$

For a theory T, $T^X = \{\varphi^X \mid \varphi \in T\}$. An ordered model M over $At(T)$ is a *stable-ordered model* of T iff $M \models T^M$, and for each subsequence M' of M, $M' \models T^M$ implies $M' = M$. With other words, M is a subsequence-minimal ordered model of T^M. The stable-ordered models of a formula φ are the ones of the theory $\{\varphi\}$.

The definition of the reduct for the classical connectives is the one for propositional theories [8, 9]. For the classical connectives and a formula φ, the context X is propagated to all direct subformulas of φ if a formula is true in X. Otherwise, φ is replaced by $\bot$. For the order operators, φ is evaluated not in X but in respective prefixes of X. An evaluation of $\Diamond\varphi$, resp., $\Box\varphi$, in X corresponds to the disjunction, resp., conjunction,

of φ with respect to all prefixes of X. Note that such a disjunction or conjunction is equivalent to $\bot$ if the formula is false in X.

Consider $X = \langle a, b\rangle$ and $\varphi = \Diamond((a \wedge \neg b) \vee b) \wedge (a \supset b)$. The reduct of φ for X is

$$\begin{aligned}
&(\Diamond((a \wedge \neg b) \vee b))^X \wedge (a \supset b)^X \\
&= \Big(\Diamond((a \wedge \neg b) \vee b)^\varepsilon \vee \Diamond((a \wedge \neg b) \vee b)^{\langle a\rangle} \vee \Diamond((a \wedge \neg b) \vee b)^{\langle a,b\rangle}\Big) \wedge \big(a^X \supset b^X\big) \\
&= (\Diamond\bot \vee \Diamond a \vee \Diamond b) \wedge (a \supset b) \\
&= (\Diamond a \vee \Diamond b) \wedge (a \supset b)
\end{aligned}$$

The single minimal ordered model of φ^X is $\langle b\rangle$, thus X is not stable. But this is reasonable: In the formation of the minimal ordered model of the reduct, subformula $\psi = \Diamond((a \wedge \neg b) \vee b)$ is not strong enough to force a (and subsequently b by $a \supset b$) when evaluated in context X as ψ can also be satisfied by $\langle b\rangle$. This is reflected by the disjunction in φ^X. It can be checked that the only stable-ordered model of φ is $\langle b\rangle$.

It holds that every stable-ordered model is also an ordered model:

Theorem 1. *For a formula $\varphi \in \mathcal{L}$ and ordered model X over $At(\varphi)$, $X \models \varphi$ iff $X \models \varphi^X$.*

Proof. For the if direction, assume that $X \not\models \varphi$. Then, $\varphi^X \equiv \bot$ and $X \not\models \varphi^X$ follows.

We show the only-if direction by induction on the structure of φ. If $\varphi = \bot$ or φ is atomic, $X \models \varphi$ implies $\varphi^X = \varphi$ and $X \models \varphi^X$ follows.

Otherwise, φ is of form $\varphi_1 \otimes \varphi_2$, $\otimes \in \{\wedge, \vee, \supset\}$, $\circ\varphi_1$, $\Diamond\varphi_1$, or $\Box\varphi_1$. The cases for the classical connectives are straight forward: For any $\otimes \in \{\wedge, \vee, \supset\}$, $\varphi^X = \varphi_1^X \otimes \varphi_2^X$. By induction, $X \models \varphi_1$ iff $X \models \varphi_1^X$ and $X \models \varphi_2$ iff $X \models \varphi_2^X$. Hence, $X \models \varphi_1 \otimes \varphi_2$ implies $X \models \varphi_1^X \otimes \varphi_2^X$, and $X \models \varphi^X$ follows.

Assume $\varphi = \circ\varphi_1$. $X \models \varphi$ implies that that there is a prefix X' of X with $|X| - |X'| = 1$ and $X' \models \varphi_1$. By the inductive hypothesis, $X' \models \varphi_1^{X'}$, and thus $X \models \circ\varphi_1^{X'}$. As $\varphi^X = \circ\varphi_1^{X'}$, $X \models \varphi^X$ follows.

Assume $\varphi = \Diamond\varphi_1$. $X \models \varphi$ implies that there is some prefix X' of X with $X' \models \varphi_1$. By induction, $X' \models \varphi_1^{X'}$. This implies that $X \models \Diamond\varphi_1^{X'}$. As $\Diamond\varphi_1^{X'}$ is a disjunct of φ^X by definition, we conclude with $X \models \varphi^X$.

Finally, assume $\varphi = \Box\varphi_1$. $X \models \varphi$ implies that each prefix X' of X satisfies φ_1 and, by induction, $\varphi_1^{X'}$. Hence, X satisfies $\Box\varphi_1^{X'}$ for each prefix X' of X, and consequently $X \models \varphi^X$. □

Our logic indeed generalises the stable-model semantics as the stable models for theories without order operators correspond to its stable-ordered models and vice versa.

As stable model existence for disjunctive programs is a Σ_2^P-complete problem [6], Σ_2^P-hardness for deciding existence of a stable-ordered model for a formula follows. Deciding if an ordered model satisfies a formula in $\mathcal{L}$ can be done in polynomial time [16].[2] This implies that checking whether an ordered model is a subsequence-minimal model of a formula is in coNP. As we can compute the reduct φ^X in polynomial time, the following result follows:

[2] Although this result was formulated for $\mathcal{L}$ without $\circ$, it is applicable for $\mathcal{L}$ mutatis mutandis.

Theorem 2. *Given a formula $\varphi \in \mathcal{L}$, the problem of deciding whether a stable-ordered model for φ exists is Σ_2^P-complete.*

4 Applications

We illustrate ASP with theories under the stable-ordered model semantics with problems involving knowledge representation, temporal reasoning, and combinatorial search.

Activtiy Reasoning. We studied *activity reasoning* based on achieving hierarchically structured goals in previous work [16]. A goal can depend on subgoals that need to be reached beforehand. An activity model is a formal description of goal hierarchies together with constraints, where the activities correspond to top-level goals. For illustration, consider the activity model for activities a and d involving the subgoals b, c, and e, where a requires b and c, d requires e, e requires c or f, and c cannot precede b. It can be formalised by the following $\mathcal{L}$ formulas:

$$a \supset \Diamond(\neg a \wedge (b \wedge c)) \tag{1}$$
$$d \supset \Diamond(\neg d \wedge e) \tag{2}$$
$$e \supset \Diamond(\neg e \wedge (c \vee f)) \tag{3}$$
$$\neg\,(b \wedge \Diamond(\neg b \wedge c)) \tag{4}$$

Formulas (1)–(3) represent the subgoal relation, and (4) is the constraint regarding the order of c and b. Assume we observe that c is already archived. We want to explain this observation in terms of the activity model. That is, does some activity entail the observation? We use the following formulas:

$$a \vee d \tag{5}$$
$$\neg\neg\Diamond(c \wedge \circ\mathbb{I}) \tag{6}$$

The single stable-ordered model of formulas (1)–(6) is $\langle c, e, d\rangle$. Only activity d can explain the observation as a can never be realised because of constraint (4). This is an example of abductive reasoning from the activities as hypotheses: Formula (5) nondeterministically selects either activity a or activity d and (6) enforces that the activity model derives the observation that c has already been archived.

Combinatorial Event-Sequence Testing. In many applications, faults are triggered by the order of events. Based on the fault model that the number of events relevant to a bug is typically low, Kuhn et al. introduced *sequence-covering arrays* (SCAs) as combinatorial designs to avoid the high costs of exercising all possible event sequences [11]. ASP for event-sequence testing has been studied in previous work [4]. Given a set E of events, an E-sequence is a permutation of the events in E. An SCA of strength t and size n is a set $\{e_1, \ldots, e_n\}$ of E-sequences such that each sequence of pairwise distinct atoms from E with length t is subsequence of some e_i, $1 \leq i \leq n$. We assume a fixed $t = 3$.

Often, some sequences are not feasible, e.g., "paste" cannot happen before "copy". Let C be a set of binary constraints over E with $(a, b) \in C$ iff a must not precede b in any e_i. Define

$$P = \{(a, b, c) \in E^3 \mid a \neq b, b \neq c, a \neq c, (a, b) \notin C, (b, c) \notin C, \text{ and } (a, c) \notin C\}$$

The following $\mathcal{L}$ formulas with parameter n encode all SCAs of size n compatible with the constraints in C:

$$\bigwedge\nolimits_{a \in E, 0<i\leq n} a_i \tag{7}$$

$$\bigwedge\nolimits_{(a,b)\in C, 0<i\leq n} \neg(b_i \wedge \Diamond(\neg b_i \wedge a_i)) \tag{8}$$

$$\bigwedge\nolimits_{(a,b,c)\in P} \bigvee\nolimits_{0<i\leq n} (c_i \wedge \Diamond(\neg c_i \wedge b_i \wedge \Diamond(\neg b_i \wedge a_i))) \tag{9}$$

Formula (7) defines the test-input space in terms of sets of E-sequences. Index i means that event a_i belongs to e_i. Formula (8) is a constraint that enforces that there is no e_i where a precedes b if $(a, b) \in C$. Set P contains all triples of events that need to be covered, i.e., occur as a subsequence of some E-sequence. Finally, coverage of all elements of P is guaranteed by Formula (9).

Fault Detection in Concurrent Programs. Finding bugs in multi-threaded programs is notoriously hard due the vast number of possible thread interleavings. A program consists of threads $t_1, \ldots, t_m$ and a set E of shared variables. Each thread t_i is modelled by a sequence $\langle a^i_1, \ldots, a^i_{n_i}\rangle$ of read or write accesses to variables from E. A thread interleaving is a total order on all a^i_j such that the relative order within the threads is preserved. Based on the fault model that many bugs are caused by reading a variable that has been defined by the wrong writer, *define-use pairs* have been studied as coverage criterion to select interesting interleavings [12]. A define-use pair $(w, r)_v$ is a write and a read access to the same variable v. An interleaving covers $(w, r)_v$ iff w precedes r, and there is no write to v inbetween.

Let P be the set of define-use pairs. To obtain total coverage of P by a set of interleavings, we iterate the following steps until $P = \emptyset$:

(i) search for an interleaving I that covers some $p \in P$, and
(ii) remove all pairs covered by I from P.

The following $\mathcal{L}$ formulas can be used to search for an interleaving that covers a given define-use pair $q = (w, r)_v$ and to identify all additionally covered ones. Let W_v be the set of all write accesses to a variable v.

$$\bigwedge\nolimits_{i=1}^{m} \left(a^i_1 \wedge \bigwedge\nolimits_{j=1}^{n_i-1} \left(a^i_{j+1} \wedge \Diamond\left(\neg a^i_{j+1} \wedge a^i_j\right)\right)\right) \tag{10}$$

$$\bigwedge\nolimits_{(w,r)_v\in P} \left(\left(r \wedge \Diamond(\neg r \wedge w) \wedge \bigwedge\nolimits_{a\in W_v\setminus\{w\}} \neg\Diamond(w \wedge a \wedge \neg r)\right) \supset c_{(w,r)_v}\right) \tag{11}$$

$$\neg\neg c_{(w,r)_v} \tag{12}$$

Formula (10) spans the search space of possible thread interleavings. Formula (11) derives $c_{(w,r)_v}$ if the define-use pair $(w,r)_v$ is covered. Finally, (12) is a constraint that prunes away all models where the specified define-use pair q is not covered.

Digital Forensics. A frequent problem in digital forensics is *file carving*, i.e., to recover fragmented files when file-table information is not available. Files are typically stored in terms of clusters but these clusters are not necessarily in order on a storage device. The problem of recovering multiple files from a set of clusters has been studied by Pal and Memon [14] as a k-vertex disjoint graph problem. The clusters are the vertices V of a graph $G = (V, E)$, some clusters are identified as headers $H \subseteq V$ or footers $F \subseteq V$, and $(a, b) \in E$ iff $a \notin F$, $b \notin H$, and the likelihood that b follows a—calculated by a suitable metric—is above a fixed threshold. We want to find k paths in G that start with a header and end in a footer such that each cluster appears in exactly one path. We can formalise this problem concisely as follows:

$$\bigwedge\nolimits_{a \in V} a \tag{13}$$

$$\bigwedge\nolimits_{(a,b) \in V^2 \setminus (E \cup (F \times H))} \neg\Diamond(b \wedge \circ(a \wedge \neg b) \wedge \neg\circ\circ a) \tag{14}$$

Formula (13) spans the search space in terms of permutations of all clusters. Paths where b follows a but $(a, b) \notin E$, unless a is a footer and b is a header, i.e., a new path starts, are excluded via (14) . Each stable-ordered model of formulas (13)–(14) describes a solution to the specified k-vertex disjoint graph problem.

5 Discussion

Our idea of sequences of atoms as models naturally lends itself to reasoning about goal achievements when goals are seen as atomic entities and the order operators have a temporal interpretation. This is by design as our initial motivation comes from activity reasoning [16]. In fact, ordered models are a compact representation of LTL traces where in each step a single new atom becomes true. Also others dealt with nonmonotonic temporal logics based on LTL [2, 3, 1, 5], but the idea of ordered models is quite unique and allows for a semantics closer to standard stable models. Notably, the complexity of deciding model existence remains in Σ_2^P. Although this is a distinctive advantage compared to aforementioned related work, the flip-side is reduced expressiveness. Yet, we demonstrate versatile applications from activity reasoning combinatorial testing, concurrent programming, and digital forensics. Although these problems can also be encoded in standard ASP, we think that dedicated order operators allow for more natural and concise problem encodings.

We expect that common results for theories under the stable-model semantics (strong equivalence, splitting sets, etc.) hold for theories under the stable-ordered model semantics as well but leave this for future work. Also, we plan to identify normal forms of theories that are closer to the familiar rule based syntax of logic programming and study translations into standard ASP so that existing solvers can be used for model generation.

References

1. Aguado, F., Cabalar, P., Diéguez, M., Pérez, G., Vidal, C.: Temporal equilibrium logic: a survey. Journal of Applied Non-Classical Logics 23(1-2), 2–24 (2013)
2. Baral, C., Zhao, J.: Non-monotonic temporal logics for goal specification. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007). pp. 236–242. AAAI Press (2007)
3. Baral, C., Zhao, J.: Non-monotonic temporal logics that facilitate elaboration tolerant revision of goals. In: Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI 2008). pp. 406–411. AAAI Press (2008)
4. Brain, M., Erdem, E., Inoue, K., Oetsch, J., Pührer, J., Tompits, H., Yilmaz, C.: Event-sequence testing using answer-set programming. International Journal on Advances in Software 5(3 & 4), 237–251 (2012)
5. Cabalar, P., Kaminski, R., Schaub, T., Schuhmann, A.: Temporal answer set programming on finite traces. TPLP 18(3-4), 406–420 (2018)
6. Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: Propositional case. Ann. Math. Artif. Intell. 15(3-4), 289–323 (1995)
7. Erdem, E., Gelfond, M., Leone, N.: Applications of answer set programming. AI Magazine 37(3), 53–68 (2016)
8. Ferraris, P.: Answer sets for propositional theories. In: Proceedings of the 8th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2005). LNCS, vol. 3662, pp. 119–131. Springer (2005)
9. Ferraris, P., Lifschitz, V.: Mathematical foundations of answer set programming. In: We Will Show Them! Essays in Honour of Dov Gabbay, Volume One. pp. 615–664. College Publications (2005)
10. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of the 5th International Conference and Symposium on Logic Programming. pp. 1070–1080. MIT Press (1988)
11. Kuhn, D.R., Higdon, J.M., Lawrence, J., Kacker, R., Lei, Y.: Combinatorial methods for event sequence testing. In: Proceedings of the 5th IEEE International Conference on Software Testing, Verification and Validation (ICST 2012). pp. 601–609. IEEE Computer Society (2012)
12. Lu, S., Jiang, W., Zhou, Y.: A study of interleaving coverage criteria. In: Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering. pp. 533–536. ACM (2007)
13. Marek, V.W., Truszczyński, M.: Stable models and an alternative logic programming paradigm. In: The Logic Programming Paradigm, pp. 375–398. Springer (1999)
14. Memon, N.D., Pal, A.: Automated reassembly of file fragmented images using greedy algorithms. IEEE Trans. Image Processing 15(2), 385–393 (2006)
15. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Ann. Math. Artif. Intell. 25(3-4), 241–273 (1999)
16. Oetsch, J., Nieves, J.C.: A knowledge representation perspective on activity theory. arXiv.org eprint arXiv:1811.05815 (2018)
17. Pearce, D.: A new logical characterisation of stable models and answer sets. In: Proceedings of Non-Monotonic Extensions of Logic Programming (NMELP 1996). LNCS, vol. 1216, pp. 57–70. Springer (1997)
18. Pnueli, A.: The temporal logic of programs. In: Proceedings of the 18th IEEE Symposium on the Foundations of Computer Science (FOCS 1977). pp. 46–57. IEEE Computer Society Press (1977)