Generalizing Multi-Context Systems for Reactive Stream Reasoning Applications*

Stefan Ellmauthaler

Intelligent Systems, Institute of Computer Science, Leipzig University
P.O. Box 100920, 04009 Leipzig, Germany
ellmauthaler@informatik.uni-leipzig.de

Abstract

In the field of artificial intelligence (AI), the subdomain of knowledge representation (KR) has the aim to represent, integrate, and exchange knowledge in order to do some reasoning about the given information. During the last decades many different KR-languages were proposed for a variety of certain applications with specific needs. The concept of a managed Multi-Context System (mMCS) was introduced to provide adequate formal tools to interchange and integrate knowledge between different KR-approaches. Another arising field of interest in computer science is the design of online applications, which react directly to (possibly infinite) streams of information. This paper presents a genuine approach to generalize mMCS for online applications with continuous streams of information. Our major goal is to find a good tradeoff between expressiveness and computational complexity.

1998 ACM Subject Classification 1.2.11 Distributed Artificial Intelligence

Keywords and phrases Knowledge Representation, Artificial Intelligence

Digital Object Identifier 10.4230/OASIcs.ICCSW.2013.19

1 Introduction

Research in the field of knowledge representation has originated a large variety of formats and languages. To use those formal concepts a wealth of tools have emerged (e.g. databases, ontologies, triple-stores, modal logics, temporal logics, nonmonotonic logics, logic programs under nonmonotonic answer set semantics, ...). Those tools were designed for specific needs of certain applications in mind. With the idea of a *"connected world"*, nowadays we do not intend to divide information over different applications. It is desirable to have all information available for every application if need be. To express all of this knowledge, represented in specifically tailored languages, in a universal language would be too hard to achieve from the point of view of complexity as well as the troubles arising from the translation of the representations.

A second issue in current knowledge representation, which is already addressed in different fields of knowledge representation (e.g. stream data processing and querying [10, 9], stream reasoning with answer set programming [6], forgetting in general [8, 5]), is the lack of *online* usage of KR tools and formalisms. Most of the approaches only assume one-shot computations, which is triggered by a user. This may be a specific request in the form of a query to a computer. In practice there are many applications where knowledge is provided in a constant flow of information and it is desired to reason over this knowledge in a continuous manner.

* This research has been funded by DFG (project FOR 1513)

2013 Imperial College Computing Student Workshop (ICCSW'13).
Editors: Andrew V. Jones, Nicholas Ng; pp. 19–26

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

ICCSW

The concept of nonmonotonic Multi-Context Systems (MCS) [2] is a promising approach to achieve a formalism which will not suffer from any of the two shortcomings of current KR-languages. The problem of connecting divided knowledge was the motivation of MCS and its successor [4]. In the following we want to generalize those mMCS to be *reactive* to their environment.

The paper proceeds as follows. After providing some motivating examples for an application of reactive managed Multi-Context Systems in Section 2, we will give an overview on the necessary background regarding managed Multi-Context Systems in Section 3. Section 4 will then introduce the new reactive concepts as an extension to managed Multi-Context Systems. A conclusion with possible future work and a discussion of related work concludes the paper.

2 Motivation

In this section we want to describe one specific application, where a reactive version of MCS would be beneficial. Although our new concept was intended to work for this special use-case, the present approach provides a general and abstract formalism for the whole variety of online-applications.

2.1 Assisted Living

In general we mean by Assisted Living some sort of intelligent apartment, which tries to analyze the behavior of its inhabitants to support them in their daily living. To be more precise, one application of Assisted Living could be the detection of emergencies that may occur in the apartment. As an example imagine a kitchen with different sensors installed. One severe emergency would be that the resident forgot to turn off the cooking stove. Then the intelligent system should react accordingly and either turn it off by itself or by giving an adequate reminder to the resident. Another example could be the detection of an accident. In case one inhabitant had a heart attack or got injured badly, the intelligent apartment should detect it and launch an appropriate emergency-measure (e.g. emergency call). Another convenience-increasing action that could be taken by the apartment is to detect whether the inhabitant wants to be disturbed or not (e.g. he is sleeping). Based on this knowledge it could become handy to mute the mobile phone to avoid an unwanted interruption. But it would be wise to enable the sound again if someone important is calling or the alarm clock wants to awake the inhabitant. These examples are only few possibilities and they may be extended by additional interaction of the apartment (e.g. by giving it access to robots and similar mechanics).

2.2 Realization

The above described apartment may be realized by the installation of different sensors in each room. Some possibilities would be: cameras, microphones, pressure plates, thermostats, and power meters. Each of these sensors will provide a constant flow of information (i.e. a stream). For one apartment an intelligent agent will reason about these streams and conclude on the current behavior of the inhabitant (e.g. if the inhabitant is in the kitchen and the cooking stove is on then he will cook something). Due to the high amount of information given by the stream of each sensor, it is now desirable to get some preprocessing done by the sensors before they send their information to the agent (e.g. the camera detects movement or identifies objects). To get more sophisticated information it is now imaginable to group

a set of sensors to an own agent with its own reasoning (e.g. all sensors in the kitchen, all sensors that track movement, ...). Then it is the task for the apartment-agent to find reasonable conclusions based on the information delivered by the different sensors/agents. Those conclusions can be the current activity of one inhabitant and the appropriate reactions by the agent itself. Due to the possibility of wrong sensor-data, previously drawn conclusions which are refuted, and other inconsistencies/conflicts between the different streams, it is now important to find some kind of equilibria between those agents in a similar way as it is described for Multi-Context Systems [3, 4, 1].

In addition, the agent may encounter many situations with exceptions. One example could be that an inhabitant is cooking and during the waiting time he goes to the restroom. In this case the apartment would not be asked for detecting an emergency. It may also not be an emergency if the inhabitant is going to watch television during the cooking time. But it is an emergency situation if he falls asleep during watching television and will not awake when the meal is done.

3 Background

In this section we will present the already existing definitions for managed Multi-Context Systems (mMCS) [4]. Intuitively, the management extension of MCS changes the bridge rules of the MCS in such a way, that the head of the bridge rule is an arbitrary operator. At first we need to define a logic suite, which allows dynamic changes of the context semantics.

▶ **Definition 1.** A *logic suite* $LS = (\mathcal{BS}_{LS}, \mathcal{KB}_{LS}, \mathcal{ACC}_{LS})$ consists of the set $\mathcal{BS}_{LS}$ of possible belief sets, the set $\mathcal{KB}_{LS}$ of well-formed knowledge-bases, and a nonempty set $\mathcal{ACC}_{LS}$ of possible semantics of LS, i.e. $ACC_{LS} \in \mathcal{ACC}_{LS}$ implies $ACC_{LS} : \mathcal{KB}_{LS} \to 2^{\mathcal{BS}_{LS}}$.

Each logic suite LS has a set of formulas $F_{LS} = \{s \in kb \mid kb \in \mathcal{KB}_{LS}\}$ which represent all formulas occurring in its knowledge base. To describe which operators are allowed, we use a *management base* OP, which is a set of operation names. For each logic suite LS and management base OP, let $F_{LS}^{OP} = \{o(s) \mid o \in OP, s \in F_{LS}\}$ be the set of operational statements which can be built from OP and F_{LS}. The semantics of statements in F_{LS}^{OP} is defined in terms of a *management function*. It allows to modify formulas in a context (e.g. by addition, removal, ...) as well as any desired operation to be applied on a formula or a context.

▶ **Definition 2.** A *management function* over a logic suite LS and a management base OP is a function $mng : 2^{F_{LS}^{OP}} \times \mathcal{KB}_{LS} \to 2^{\mathcal{KB}_{LS} \times \mathcal{ACC}_{LS}} \setminus \{\emptyset\}$.

▶ **Definition 3.** A *managed Multi-Context System* M is a collection $(C_1, \ldots, C_n)$ of managed contexts where, for $1 \leq i \leq n$, each managed context C_i is a quintuple $C_i = (LS_i, kb_i, br_i, OP_i, mng_i)$ such that

- $LS_i = (\mathcal{BS}_{LS_i}, \mathcal{KB}_{LS_i}, \mathcal{ACC}_{LS_i})$ is a logic suite,
- $kb_i \in \mathcal{KB}_{LS_i}$ is a knowledge base,
- OP_i is a management base,
- br_i is a set of bridge rules for C_i, with the form

$$op_i \leftarrow (c_1 : p_1), \ldots, (c_j : p_j), not(c_{j+1} : p_{j+1}), \ldots, not(c_m : p_m).$$

  such that $op_i \in F_{LS_i}^{OP_i}$ and for all $1 \leq k \leq m$ there exists a context $c_k \in (C_1, \ldots, C_n)$ such that $p_k \in S \in \mathcal{BS}_{LS_{c_k}}$, and
- mng_i is a management function over LS_i and OP_i.

For a bridge rule $r \in br_i$ we will use $op(r)$ to denote the operator $op_i \in F_{LS_i}^{OP_i}$ and $body(r)$ denotes the set $\{(c_{k_1} : p_{k_1}) \mid 1 \leq k_1 \leq j\} \cup \{not(c_{k_2} : p_{k_2}) \mid j < k_2 \leq m\}$.

A *belief state* $S = (S_1, \ldots, S_n)$ of M is a belief set for every context, i.e. $S_i \in \mathcal{BS}_{LS_i}$. We denote the set of applicable operations by $app_i(S) = \{op(r) \mid r \in br_i \wedge S \models body(r)\}$. The term of *equilibrium* is used to define the semantics of an mMCS.

▶ **Definition 4.** Let $M = (C_1, \ldots, C_n)$ be an mMCS. A belief state $S = (S_1, \ldots, S_n)$ is an equilibrium of M iff for every $1 \leq i \leq n$ there exists some $(kb_i', ACC_{LS_i}) \in mng_i(app_i(S), kb_i)$ such that $S_i \in ACC_{LS_i}(kb_i')$.

4 Reactive Managed Multi-Context Systems

In the following we will present different approaches to a reactive managed Multi-Context System. At first we will try to get a generalization of the already existing approach of managed Multi-Context Systems. Afterwards we will propose a less complex approach for faster reactions to incoming information. Finally we will combine both variants to gain a solution which benefits from both approaches.

4.1 Preference-Based Iterative Managed Multi-Context System

This part will sketch what needs to be added to the current approach of mMCS to make it suitable for the previously given applications. We have chosen a similar approach to the reactive concept as it was applied by Schaub et al. [6, 7] for their Answer Set Programming-Solver. So we will manipulate our knowledge bases iteratively, based on the current equilibria which may take different input stream information into account. Therefore we will refer to it as an *iterative managed Multi-Context System (imMCS)*. For easier recognition of the different tasks, we will introduce different types of contexts. We will need at least three of them:

- *observing contexts:* these contexts are connected via sensors to the outside world and obtain new information constantly.
- *reasoning contexts:* these contexts are internal modules. It is important that those are not connected to sensors and so they do rely on the information given by other contexts. They are responsible to interpret the different observations and determine what is going on, i.e. are things working properly, does an action need to be taken.
- *control contexts:* this context has the role to do some kind of meta-reasoning for the imMCS. Its role is to:
  1. set sliding windows for other contexts[1],
  2. set inconsistency handling policies (e.g. take sensor reliability into account in case of inconsistencies),
  3. set the used semantics and reasoning modes,
  4. determine necessary actions[2] (e.g. start an alarm), and
  5. decide which contexts need to re-reason (e.g. after a change done by the control context) or which context shall be idle.

To model the dynamic development of equilibria over time, we introduce the notion of a *run of an mMCS*. Intuitively a run describes the provided knowledge and the computed equilibria at a given time.

[1] Reactive reasoning mechanisms use sliding windows to handle possibly infinite streams (c.f. [6]).
[2] It would be reasonable to use such control contexts as the way to communicate with the real world

▶ **Definition 5.** Let M be a managed MCS with contexts $C = (C_1, \ldots, C_n)$ ($C_1, \ldots, C_k$ are observer contexts). Let $Obs = (Obs^0, Obs^1, \ldots)$ be a sequence of observations, that is, for $j \geq 0$, $Obs^j = (Obs_i^j)_{i \leq k}$, where Obs_i^j is the new (sensor) information for context i at step j, which is formalized as sets of formulas. A run R of M induced by Obs is a sequence

$$R = Kb^0, Eq^0, Kb^1, Eq^1, \ldots$$

where

- $Kb^0 = (Kb_i^0)_{i \leq n}$ is the collection of initial knowledge bases, Eq^0 an equilibrium of Kb^0,
- for $j \geq 1$ and $i \leq n$, Kb_i^j is the knowledge base of context C_i produced by the context's management function for the computation of Eq^{j-1}, and $Kb^j = (Kb_i^j)_{i \leq n}$,
- for $j \geq 1$, Eq^j is an equilibrium for the knowledge bases

$$(Kb_0^j \cup Obs_0^j, \ldots, Kb_k^j \cup Obs_k^j, Kb_{k+1}^j, \ldots, Kb_n^j).$$

We call $M' = (C, Obs)$ an *iterative managed MCS* (imMCS).

Note that there may be more than one equilibrium, which would lead to different knowledge bases at the next step of the run. To avoid this multiplication of underlying knowledge, we need to introduce a method to reduce the number of possible equilibria. For this task there are different possible approaches:

1. usage of brave and cautious reasoning methods[3] for the selection of the applicable operations on the contexts.
2. preferences over the bridge rules to get a preferred equilibrium.

In general it may lead to side effects when using brave reasoning for the selection of applicable operations. For example one equilibrium may add a positive literal to a knowledge base, while another equilibrium would add the negated literal. That would lead obviously to an inconsistency although both equilibria were consistent. On the other hand cautious reasoning may lead to a situation where some crucial consistency preserving operations may be missing (c.f. Example 6).

▶ **Example 6.** Let Kb_1^0 be an initial knowledge base in an imMCS M'. The operation *insert* (resp. *revoke*) adds (resp. removes) formulas to (resp. from) the knowledge base. Suppose the negated literals $\neg a$ and $\neg b$ are both in Kb_1^0. The computation of the equilibria results in two sets of belief states $\{Eq_1^0, Eq_2^0\} = Eq^0$, where $app_1(Eq_1^0) = \{revoke(\neg a), insert(a \vee b)\}$ and $app_1(Eq_2^0) = \{revoke(\neg b), insert(a \vee b)\}$. With cautious reasoning only the operation $insert(a \vee b)$ would be executed, which would result in an inconsistent knowledge base.

To ensure that only one equilibrium remains, we will introduce the preference function $pref_i$. Each context provides this function, which takes the set of equilibria and returns a strict total order over them. In addition the whole Multi-Context System provides the preference function *pref*, which takes the total orderings and returns one unique equilibrium.

▶ **Definition 7.** Let $M = (C, Obs)$ be an imMCS and $\mathcal{EQ}$ be a set of equilibria. $M_p = (C, Obs, pref)$ is a preference based imMCS *(pimMCS)* where

- each context C_i has a function $\text{pref}_i : \mathcal{EQ} \to total(\mathcal{EQ})$, where $total(\mathcal{EQ}) = \{R \subseteq \mathcal{EQ} \times \mathcal{EQ} \mid R \text{ is strictly totally ordered}\}$ to associate a strict total ordering of equilibria to each context, and
- the function $pref : (pref_1(\mathcal{EQ}), \ldots, pref_n(\mathcal{EQ})) \mapsto Eq$ returns exactly one equilibrium $Eq \in \mathcal{EQ}$.

[3] Intuitively, given alternative sets of beliefs, for brave reasoning it is sufficient that one belief set supports a conclusion, while cautious reasoning requires that each belief set supports a conclusion

Intuitively, each context propagates its most appreciated equilibria. Afterwards the Multi-Context System determines the "best" fitting equilibrium. Note that at this point we do not intend to give a semantic definition for those two functions. How these equilibria are ordered and how the equilibrium is selected remains adjustable to the specific instance of the Multi-Context System.

4.2 Reactive Bridge Rules

In general the computation of equilibria is expensive [4]. It was shown that the identification of a global equilibria is always one level higher on the polynomial hierarchy than the computation of belief sets of the context with the hardest problem. Due to this potentially high amount of computation time, we present another approach, which will not utilize the concept of global equilibria. The intuitive idea behind our *Reactive Bridge Rules* (RBR) is to provide rules to add supplementary information to the input stream of another reactive context. These rules are evalutated over the belief sets of the different contexts. To control how "informative" one of those rules is, it can be specified for each rule whether its literals need to occur in one or every belief set of the context.

▶ **Definition 8.** A *Reactive Bridge Rule* (RBR) r for a context C_i of a collection of n contexts is a rule of the form

$$t, j : h \leftarrow b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m$$

where

- $t \in \{b, c\}$ specifies whether the literals need to be evaluated bravely or cautiously,
- $j \leq n$ specifies which context will be provided with additional information,
- h is some information which may be added to the input stream of C_j, and
- for $l \leq m$, b_l is a literal.

We will denote the body of one RBR r as *body(r)*, all positive literals b_l, where $l \leq k$ as $b^+(r)$, and all negated literals b_l, where $k < l \leq m$ as $b^-(r)$. Based on the given evaluation mode of the rule, there are different semantics to be applied to the rule. Note that it is obligatory for each context that it has an input stream.

▶ **Definition 9.** Let r be an RBR of a context C_i, $ACC_{LS_i} \in \mathcal{ACC}_{LS_i}$ be a selected semantics, and $S = \{S_1 \ldots, S_j\}$ be the belief sets of C_i at the step t, such that $S = ACC_{LS_i}(kb_i^t)$, where kb_i^t is the knowledge base of context C_i at step t.

- If r is a cautious RBR, it is satisfied if $\forall_{B \in S}(b^+(r) \subseteq B \wedge b^-(r) \cap B = \emptyset)$.
- If r is a brave RBR, it is satisfied if $\exists_{B \in S}(b^+(r) \subseteq B \wedge b^-(r) \cap B = \emptyset)$.

If a rule r is satisfied, then h will be added to the input stream of the context C_j at step $t + 1$.

We will write $\mathcal{RBR}_i^j$ to denote the set of added information to the input stream of context i at step $j+1$, based on the belief sets of step j. Intuitively a RBR wants to inform other contexts of the outcome of different conclusions drawn by a context, based on its observations. The two types of rules were chosen to distinguish between possible conclusions which may be very important and those conclusions which can be drawn safely. In our assisted living scenario there may be events which are more critical than others. For example the possibility of an emergency should be considered as soon as possible, even if it is not assured in every belief set of a context. On the other hand some conclusions may not be neccessary to be forwarded to another context. One example could be the control of the door lock. The door should only open for visitors if every belief set is sure that the person may enter the assisted living environment.

4.3 Combination Of Both Concepts

The two newly introduced concepts have their advantages and disadvantages. The pimMCS do compute equilibria and therefore it is required that all involved contexts agree on a decision. Alas, their computation is quite expensive. Thus it may happen that the computation of an equilibrium takes very long compared to the intervals of newly arriving information in the input streams. With infinite data streams in mind this is a serious issue. The use of sliding windows will force that older, but probably important information is lost due to its size. In case the window is extended automatically to be able to fit all new information since the last equilibria-computation in the run, this memory will grow larger the longer the computation takes. In addition a larger window may also increase the time effort of the next computation of the equilibria, which is some kind of a vicious cycle.

On the other hand RBRs only need the belief sets of each context and there is no need for any agreement on their conclusions and beliefs. This computation involves no communication between the contexts and further it is not neccessary to find an equilibrium. Of course the results are not as strong as an equilibrium, as there is no commonly acceptable belief set of the problem and only local points of view about them.

Now we want to combine both approaches to achieve a *Reactive Managed Multi-Context System (rmMCS)*, which takes the advantages of both ideas and avoids their disadvantages. In general it is desirable to get a formal system which computes equilibria on which decisions are done. Our idea is to compute a run for a pimMCS, where each context has an input stream. During the computation of an equilibrium each context can agree with, RBRs are allowed to manipulate the input streams of the contexts. We also allow each context to change its belief sets based on new stream information, such that another set of RBRs is allowed to manipulate the streams further. Note that this manipulation and change of beliefs shall not affect the computation of the equilibrium. Intuitively, it can be seen as a parallel process.

▶ **Definition 10.** Let M be a preference based iterative managed MCS with contexts $C_0, \ldots, C_n$, context-specific preferences $pref_0, \ldots, pref_n$ and a global preference $pref$. Let $Is = (Is^0, Is^1, \ldots)$ be a sequence of input streams, that is, for $j \geq 0$, $Is^j = (Is_i^j)_{i \leq n}$, where Is_i^j is the current input stream for context i at step j. Let $f(Eq^i)$ be a function that returns the step where the computation of the equilibrium of step i finished, and RBR be a set of reactive bridge rules. A run R of M induced by IS is a sequence

$$R = Kb^0, Eq^0, Kb^{f(Eq^0)}, Eq^{f(Eq^0)}, \ldots$$

where

- $Kb^0 = (Kb_i^0)_{i \leq n}$ is the collection of initial knowledge bases, Eq^0 an equilibrium of Kb^0,
- for $i \leq n$, $mIs_i^0 = Is_i^0$ is the modified initial input stream,
- for $j \geq 1$ and $i \leq n$, $mIs_i^j = Is_i^j \cup \mathcal{RBR}_i^{j-1}$ is the modified input stream at step j,
- for $j \geq 0$, Eq^j is the preferred equilibrium at step j,
- for $j \geq 1$ and $i \leq n$, Kb_i^j is the knowledge base of context C_i produced by the context's management function for the computation of Eq^k, such that $f(Eq^k) = j$, and $Kb^j = (Kb_i^j)_{i \leq n}$, and
- for $j \geq 1$, Eq^j is an equilibrium for the knowledge bases

$$(Kb_0^j \cup mIs_0^j, \ldots, Kb_n^j \cup mIs_n^j).$$

We call $M' = (M, Is, RBR)$ a reactive managed Multi-Context System.

5 Conclusion & Future Work

In this paper we have presented two generalizations for managed Multi-Context Systems, which can utilize streams containing information. We want to mention again that we had stream reasoners, such as `oclingo`[6] as contexts in mind. However, the presented formalism may work well with different approaches. Our goal with this new formalism is to provide a framework where as many formalisms as possible may be used as contexts.

Intended future work is an instantiation of the formalism, to model the given application of assisted living. In addition it will be necessary to investigate possible side effects of the RBR with respect to the rmMCS. In this field there are also some questions which are not answered in this paper (e.g. how should the preference functions be handled). Another interesting field is in general the usage of other formalisms. Are there any undesired effects if we use e.g. C-SPARQL [9]. Is it important to restrict our systems in any way to such that their underlying contexts are not affected in an undesired way. Additionally it is open on how to utilize KR formalisms and tools which do not support online reasoning. Is it sufficient to provide some kind of reactive add-on, which communicates between a stream and a *one-shot offline* formalism?

References

1 Gerhard Brewka. Multi-context systems: Specifying the interaction of knowledge bases declaratively. In Markus Krötzsch and Umberto Straccia, editors, *RR*, volume 7497 of *Lecture Notes in Computer Science*, pages 1–4. Springer, 2012.

2 Gerhard Brewka and Thomas Eiter. Equilibria in heterogeneous nonmonotonic multi-context systems. In *AAAI*, pages 385–390. AAAI Press, 2007.

3 Gerhard Brewka, Thomas Eiter, and Michael Fink. Nonmonotonic multi-context systems: A flexible approach for integrating heterogeneous knowledge sources. In Marcello Balduccini and Tran Cao Son, editors, *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, volume 6565 of *Lecture Notes in Computer Science*, pages 233–258. Springer, 2011.

4 Gerhard Brewka, Thomas Eiter, Michael Fink, and Antonius Weinzierl. Managed multi-context systems. In Toby Walsh, editor, *IJCAI*, pages 786–791. IJCAI/AAAI, 2011.

5 Fu-Leung Cheng, Thomas Eiter, Nathan Robinson, Abdul Sattar, and Kewen Wang. Lp-forget: A system of forgetting in answer set programming. In Abdul Sattar and Byeong Ho Kang, editors, *Australian Conference on Artificial Intelligence*, volume 4304 of *Lecture Notes in Computer Science*, pages 1101–1105. Springer, 2006.

6 Martin Gebser, Torsten Grote, Roland Kaminski, Philipp Obermeier, Orkunt Sabuncu, and Torsten Schaub. Stream reasoning with answer set programming: Preliminary report. In Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, editors, *KR*. AAAI Press, 2012.

7 Martin Gebser, Orkunt Sabuncu, and Torsten Schaub. An incremental answer set programming based system for finite model computation. *AI Communications*, 24(2):195–212, 2011.

8 Jérôme Lang and Pierre Marquis. Reasoning under inconsistency: A forgetting-based approach. *Artif. Intell.*, 174(12-13):799–823, 2010.

9 Danh Le-Phuoc, Josiane Xavier Parreira, and Manfred Hauswirth. Linked stream data processing. In Thomas Eiter and Thomas Krennwallner, editors, *Reasoning Web*, volume 7487 of *Lecture Notes in Computer Science*, pages 245–289. Springer, 2012.

10 Carlo Zaniolo. Logical foundations of continuous query languages for data streams. In Pablo Barceló and Reinhard Pichler, editors, *Datalog*, volume 7494 of *Lecture Notes in Computer Science*, pages 177–189. Springer, 2012.