Multi-Context Reasoning in Continuous Data-Flow Environments[1]

Modelling with reactive Multi-Context Systems

Stefan Ellmauthaler
ellmauthaler@informatik.uni-leipzig.de

Computer Science Institute
Leipzig University
Germany

Stream Reasoning Workshop 2019
Linköping
April, 16th 2019

1 [Ellmauthaler, 2019, Ellmauthaler, 2018, Brewka et al., 2018]

Multi-Context Systems at SR-Workshops

Berlin 2016

- Inconsistency Management in reactive Multi-Context Systems
- Stream Packing in asynchronous Multi-Context Systems (given by Jörg Pührer)

Zürich 2018

- Asynchronous Multi-Context Systems

Today

- Short introduction to reactive Multi-Context Systems
- Modelling with reactive Multi-Context Systems

Logic

an Abstract Representation

- An abstract way to define a Logic
- Capable of realising monotone and non-monotone logics
- Representing different number of values
  (e.g. binary, many valued, fuzzy values, . . .)

Definition (Logic [Brewka and Eiter, 2007])

A logic is a triple $L = \langle KB, BS, \mathbf{acc} \rangle$, where

- KB is a set of knowledge bases,
- BS is a set of belief sets, and
- $\mathbf{acc} : KB \mapsto 2^{BS}$, the *acceptance function* is a function which assigns to each knowledge base a set of belief sets.

Represent KRR Formalisms

Description Logic $\mathcal{AL}$

$L_d = \langle KB_d, BS_d, \mathbf{acc}_d \rangle$

- KB_d are all ontologies
- BS_d is the set of deductively closed subsets in $\mathcal{AL}$
- $\mathbf{acc}_d$ is a mapping of $kb \in KB_d$ to $M \subseteq 2^{BS_d}$, s.t. $\forall_{m \in M} kb \models m$ holds.

Represent KRR Formalisms

Description Logic $\mathcal{AL}$

$L_d = \langle KB_d, BS_d, \mathbf{acc}_d \rangle$

- KB_d are all ontologies
- BS_d is the set of deductively closed subsets in $\mathcal{AL}$
- $\mathbf{acc}_d$ is a mapping of $kb \in KB_d$ to $M \subseteq 2^{BS_d}$, s.t. $\forall_{m \in M} kb \models m$ holds.

Answer Set Programming

$L_{asp} = \langle KB_{asp}, BS_{asp}, \mathbf{acc}_{asp} \rangle$

- Let A be the set of all possible ground atoms
- KB_{asp} is the set of all answer set programs over A.
- $BS_{asp} = 2^A$
- $\mathbf{acc}_{asp}$ maps each ASP program to its answer sets

Origins

from mMCS via [r/e]MCS to rMCS

reactive Multi-Context Systems

- based on managed Multi-Context Systems [Brewka et al., 2011]
- old version got presented at ECAI 2014 [Brewka et al., 2014]
- evolving Multi-Context Systems at ECAI 2014 [Gonçalves et al., 2014]

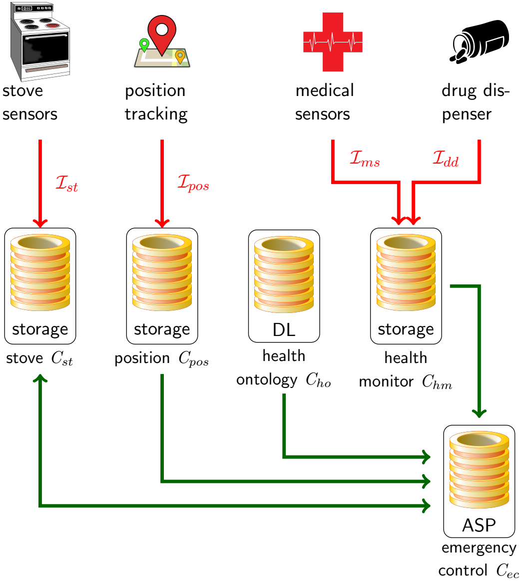
⇒ complete redefinition of rMCS

Current reactive Multi-Context Systems

- less complicated, cycle-free definitions
- a generalisation of managed Multi-Context Systems
- declarative and operative bridge rules
- results on inconsistency management
- results on complexity
- results on simulating other approaches

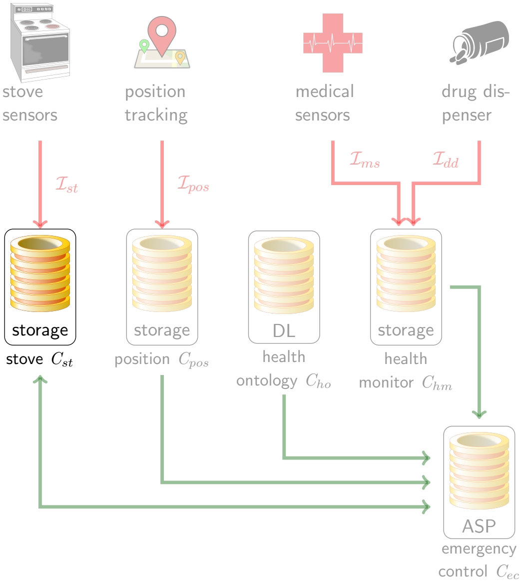
Syntax

Building Blocks



Syntax

Building Blocks



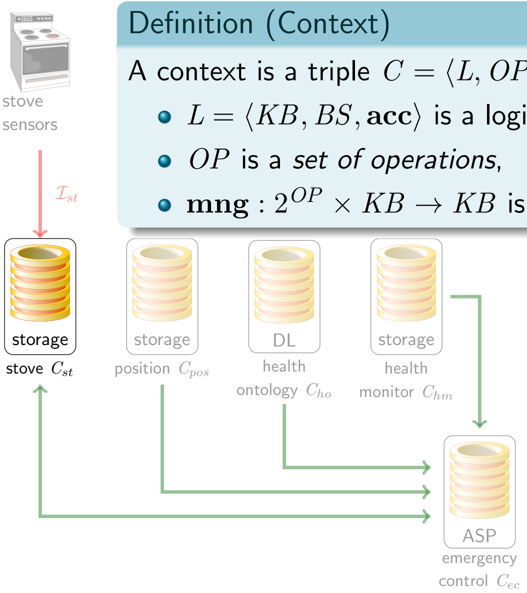
Syntax

Building Blocks

Definition (Context)

A context is a triple $C = \langle L, OP, \mathbf{mng} \rangle$ where

- $L = \langle KB, BS, \mathbf{acc} \rangle$ is a logic,
- OP is a *set of operations*,
- $\mathbf{mng} : 2^{OP} \times KB \to KB$ is a *management function*.



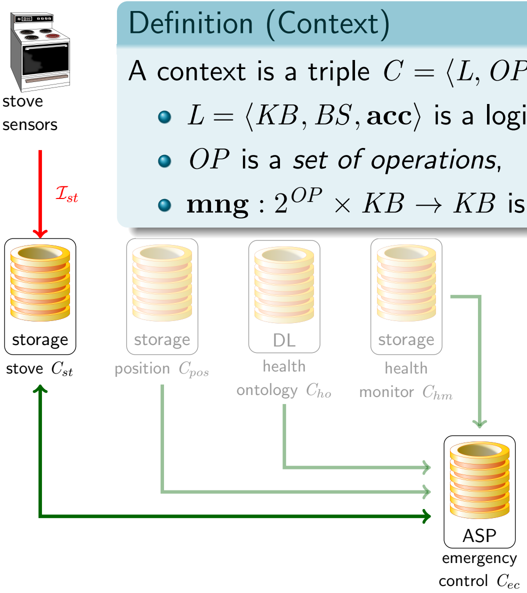
Syntax

Building Blocks

Definition (Context)

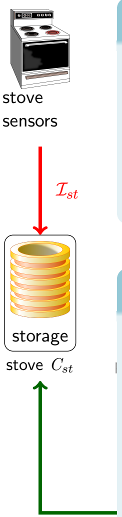
A context is a triple $C = \langle L, OP, \mathbf{mng} \rangle$ where

- $L = \langle KB, BS, \mathbf{acc} \rangle$ is a logic,
- OP is a *set of operations*,
- $\mathbf{mng} : 2^{OP} \times KB \rightarrow KB$ is a *management function*.



Syntax

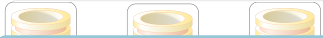
Building Blocks



Definition (Context)

A context is a triple $C = \langle L, OP, \mathbf{mng} \rangle$ where

- $L = \langle KB, BS, \mathbf{acc} \rangle$ is a logic,
- OP is a *set of operations*,
- $\mathbf{mng} : 2^{OP} \times KB \rightarrow KB$ is a *management function*.



Definition (Bridge Rule)

Let C $= \langle C_1, \dots, C_n \rangle$ be a tuple of contexts and IL $= \langle IL_1, \dots, IL_k \rangle$ a tuple of input languages. A bridge rule for C_i over C and IL, $i \in \{1, \dots, n\}$, is of the form

$$op \leftarrow a_1, \dots, a_j, \mathbf{not}\ a_{j+1}, \dots, \mathbf{not}\ a_m \text{ or}$$

$$\mathbf{next}(op) \leftarrow a_1, \dots, a_j, \mathbf{not}\ a_{j+1}, \dots, \mathbf{not}\ a_m$$

control C_{ec}

Syntax

Building Blocks

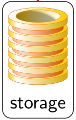
Definition (Context)

Example

$$\text{setTemp}(hot) \leftarrow st{::}\text{tmp}(T), 42 < T$$

$$\mathbf{next}(\text{setPower}(off)) \leftarrow ec{:}\text{turnOff}(stove)$$

$$\mathbf{next}(\text{setPower}(off)) \leftarrow st{::}\text{switch}, st{:}\text{pw}$$



Definition (Bridge Rule)

Let C $= \langle C_1, \dots, C_n \rangle$ be a tuple of contexts and IL $= \langle IL_1, \dots, IL_k \rangle$ a tuple of input languages. A bridge rule for C_i over C and IL, $i \in \{1, \dots, n\}$, is of the form

$$op \leftarrow a_1, \dots, a_j, \mathbf{not}\ a_{j+1}, \dots, \mathbf{not}\ a_m \text{ or}$$

$$\mathbf{next}(op) \leftarrow a_1, \dots, a_j, \mathbf{not}\ a_{j+1}, \dots, \mathbf{not}\ a_m$$

control C_{ec}

Syntax

Definition (Reactive Multi-Context System)

A *reactive Multi-Context System* is a tuple $M = \langle \mathsf{C}, \mathsf{IL}, \mathsf{BR} \rangle$, where

- $\mathsf{C} = \langle C_1, \ldots, C_n \rangle$ is a tuple of contexts;
- $\mathsf{IL} = \langle IL_1, \ldots, IL_k \rangle$ is a tuple of input languages;
- $\mathsf{BR} = \langle BR_1, \ldots, BR_n \rangle$ is a tuple such that each BR_i, $i \in \{1, \ldots, n\}$, is a set of bridge rules for C_i over C and IL.

Semantics

Current Snapshot

Definition (Configuration of Knowledge Bases)

Let $M = \langle \mathsf{C}, \mathsf{IL}, \mathsf{BR} \rangle$ be an rMCS, such that $\mathsf{C} = \langle C_1, \ldots, C_n \rangle$. A configuration of knowledge bases for M is a tuple $\mathsf{KB} = \langle kb_1, \ldots, kb_n \rangle$, such that $kb_i \in KB_i$, for each $i \in \{1, \ldots, n\}$. We use Con_M to denote the set of all configurations of knowledge bases for M.

Definition (Belief State)

Let $M = \langle \langle C_1, \ldots, C_n \rangle, \mathsf{IL}, \mathsf{BR} \rangle$ be an rMCS. Then, a belief state for M is a tuple $\mathsf{B} = \langle B_1, \ldots, B_n \rangle$ such that $B_i \in BS_i$, for each $i \in \{1, \ldots, n\}$. We use Bel_M to denote the set of all belief states for M.

Definition (Input)

Let $M = \langle \mathsf{C}, \langle IL_1, \ldots, IL_k \rangle, \mathsf{BR} \rangle$ be an rMCS. Then an input for M is a tuple $\mathsf{I} = \langle I_1, \ldots, I_k \rangle$ such that $I_i \subseteq IL_i$, $i \in \{1, \ldots, k\}$. The *set of all inputs for* M is denoted by Inp_M.

Semantics

- Only utilise Declarative Bridge Rules
- A belief state is an Equilibrium if
  - the updated knowledge base
    (i.e. the management function result on the belief state, the input, and the current configuration)
  - has as the belief state one of the accepted belief states
    (i.e. it is part of the deductive closure of the semantics)

Semantics

One-Shot Reasoning

- Only utilise Declarative Bridge Rules
- A belief state is an Equilibrium if
  - the updated knowledge base (i.e. the management function result on the belief state, the input, and the current configuration)
  - has as the belief state one of the accepted belief states (i.e. it is part of the deductive closure of the semantics)

Definition (Equilibrium)

Let $M = \langle\langle C_1, \ldots, C_n\rangle, \mathsf{IL}, \mathsf{BR}\rangle$ be an rMCS, $\mathsf{KB} = \langle kb_1, \ldots, kb_n\rangle$ a configuration of knowledge bases for M, and I an input for M. Then, a belief state $\mathsf{B} = \langle B_1, \ldots, B_n\rangle$ for M is an equilibrium of M given KB and I if, for each $i \in \{1, \ldots, n\}$, we have that

$$B_i \in \mathbf{acc}_i(kb'), \text{ where } kb' = \mathbf{mng}_i(\mathbf{app}_i^{now}(\mathsf{I}, \mathsf{B}), kb_i).$$

Semantics

- Extend the concept of the Input, to be an Input Stream
- Operative Bridge Rules allow configuration changes
- Updates are based on the previously computed Equilibrium
- Results represented as Equilibria Stream and its dual Configuration Stream

Semantics

Reactive Reasoning

Definition (Update Function)

Let $M = \langle \mathsf{C}, \mathsf{IL}, \mathsf{BR} \rangle$ be an rMCS such that $\mathsf{C} = \langle C_1, \ldots, C_n \rangle$, $\mathsf{KB} = \langle kb_1, \ldots, kb_n \rangle$ a configuration of knowledge bases for M, I an input for M, and B a belief state for M.
Then, $\mathbf{upd}_M(\mathsf{KB}, \mathsf{I}, \mathsf{B}) = \langle kb'_1, \ldots, kb'_n \rangle$ is the update function for M, such that for each $i \in \{1 \ldots, n\}$, $kb'_i = \mathbf{mng}_i(\mathbf{app}_i^{next}(\mathsf{I}, \mathsf{B}), kb_i)$ holds.

Definition (Input Stream)

Let $M = \langle \mathsf{C}, \mathsf{IL}, \mathsf{BR} \rangle$ be an rMCS such that $\mathsf{IL} = \langle IL_1, \ldots, IL_k \rangle$. An input stream for M (until τ) is a function $\mathcal{I} : [1..\tau] \to \mathsf{Inp}_M$ where $\tau \in \mathbb{N} \cup \{\infty\}$.

Semantics

Equilibria Stream

Definition (Equilibria Stream)

Let $M = \langle \mathsf{C}, \mathsf{IL}, \mathsf{BR} \rangle$ be an rMCS, KB a configuration of knowledge bases for M, and $\mathcal{I}$ an input stream for M until τ where $\tau \in \mathbb{N} \cup \{\infty\}$. Then, an equilibria stream of M given KB and $\mathcal{I}$ is a function $\mathcal{B} : [1..\tau] \rightarrow \mathsf{Bel}_M$ such that

- $\mathcal{B}^t$ is an equilibrium of M given $\mathcal{KB}^t$ and $\mathcal{I}^t$, where $\mathcal{KB}^t$ is inductively defined as
  - $\mathcal{KB}^1 = \mathsf{KB}$
  - $\mathcal{KB}^{t+1} = \mathbf{upd}_M(\mathcal{KB}^t, \mathcal{I}^t, \mathcal{B}^t)$.

In a dual manner, we will refer to the function $\mathcal{KB} : [1..\tau] \rightarrow \mathsf{Con}_M$ as the *configurations stream of M given* KB, *$\mathcal{I}$, and $\mathcal{B}$*.

Modelling Aspects

Simple Tasks

- Flipping data (self-dependent)
- Handling time
- Windows
- Forgetting

Declarative and Operational Bridge Rules

Example

Flip the power for the stove if a switch is pressed.

Declarative and Operational Bridge Rules

Example

Flip the power for the stove if a switch is pressed.

Declarative approach

- setPower$(\mathit{off}) \leftarrow st{::}\mathrm{switch}, st{:}\mathrm{pw}$
- setPower$(on) \leftarrow st{::}\mathrm{switch}, \mathbf{not}\ st{:}\mathrm{pw}$

Declarative and Operational Bridge Rules

Example

Flip the power for the stove if a switch is pressed.

Declarative approach

- setPower$(\mathit{off}) \leftarrow st{::}\mathrm{switch}, st{:}\mathrm{pw}$
- setPower$(on) \leftarrow st{::}\mathrm{switch}, \mathbf{not}\ st{:}\mathrm{pw}$
- No Equilibrium can be found

Declarative and Operational Bridge Rules

Example

Flip the power for the stove if a switch is pressed.

Declarative approach

- setPower$(\mathit{off}) \leftarrow st{::}\mathrm{switch}, st{:}\mathrm{pw}$
- setPower$(\mathit{on}) \leftarrow st{::}\mathrm{switch}, \mathbf{not}\ st{:}\mathrm{pw}$
- No Equilibrium can be found

Operational approach

- $\mathbf{next}($setPower$(\mathit{off})) \leftarrow st{::}\mathrm{switch}, st{:}\mathrm{pw}$
- $\mathbf{next}($setPower$(\mathit{on})) \leftarrow st{::}\mathrm{switch}, \mathbf{not}\ st{:}\mathrm{pw}$

Declarative and Operational Bridge Rules

Example

Flip the power for the stove if a switch is pressed.

Declarative approach

- setPower$(\mathit{off}) \leftarrow st$::switch$, st$:pw
- setPower$(on) \leftarrow st$::switch$,$ **not** st:pw
- No Equilibrium can be found

Operational approach - without sensor data

- add$(switchpower) \leftarrow st$::switch
- **next**(setPower$(\mathit{off})) \leftarrow st$:switchpower$, st$:pw
- **next**(setPower$(on)) \leftarrow st$:switchpower$,$ **not** st:pw

Handling Time

Possible ways

- Sensor
- Time-Context

Time Context

$$\begin{aligned}
\mathsf{setTime}(\mathrm{now}(\mathit{0})) &\leftarrow \mathbf{not}\ \mathit{clock}{:}\mathit{timeAvailable}\\
\mathbf{next}(\mathit{add}(\mathrm{timeAvailable})) &\leftarrow \mathit{clock}{:}\mathrm{now}(\mathit{0})\\
\mathbf{next}(\mathit{setTime}(\mathrm{now}(T+\mathit{1}))) &\leftarrow \mathit{clock}{:}\mathrm{now}(T)
\end{aligned}$$

Forgetting and Windowing

Volatile Information and Reasoning with a Window

$$\mathbf{next}(\mathsf{add}(\mathrm{alert}(\mathit{stove}, T))) \leftarrow c{::}\mathrm{now}(T), ec{:}\mathrm{alert}(\mathit{stove}).$$

$$\mathbf{next}(\mathsf{del}(\mathrm{alert}(\mathit{stove}, T))) \leftarrow stE{:}\mathrm{alert}(\mathit{stove}, T), \mathbf{not}\ ec{:}\mathrm{alert}(\mathit{stove}).$$

$$\mathsf{add}(\mathrm{emergency}(\mathit{stove})) \leftarrow c{::}\mathrm{now}(T), ec{:}\mathrm{alert}(\mathit{stove}), stE{:}\mathrm{alert}(\mathit{stove}, T'), stE{:}\mathrm{winE}(Y), |T - T'| \geq Y.$$

Dynamic Window

$$\mathbf{next}(\mathsf{set}(\mathrm{win}(P, X))) \leftarrow ed{:}\mathrm{defWin}(P, X), \mathbf{not}\ ed{:}\mathrm{susp}(E).$$

$$\mathbf{next}(\mathsf{set}(\mathrm{win}(P, Y))) \leftarrow ed{:}\mathrm{rel}(P, E, Y), ed{:}\mathrm{susp}(E).$$

$$\mathsf{alarm}(E) \leftarrow ed{:}\mathrm{conf}(E).$$

$$\mathbf{next}(\mathrm{add}(\mathrm{P}(T))) \leftarrow c{::}\mathrm{now}(T), s{::}P.$$

$$\mathbf{next}(\mathrm{del}(\mathrm{P}(T'))) \leftarrow ed{:}\mathrm{P}(T'), c{::}\mathrm{now}(T), ed{:}\mathrm{win}(P, Z), T' < T - Z.$$

Thank you for your interest

References I

... a selection

Brewka, G. and Eiter, T. (2007).
Equilibria in heterogeneous nonmonotonic multi-context systems.
In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI 2007)*, pages 385–390. AAAI Press.

Brewka, G., Eiter, T., Fink, M., and Weinzierl, A. (2011).
Managed multi-context systems.
In Walsh, T., editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, pages 786–791. IJCAI/AAAI.

Brewka, G., Ellmauthaler, S., Gonçalves, R., Knorr, M., Leite, J., and Pührer, J. (2018).
Reactive multi-context systems: Heterogeneous reasoning in dynamic environments.
Artificial Intelligence, 256:68–104.

References II

... a selection

Brewka, G., Ellmauthaler, S., and Pührer, J. (2014).
Multi-context systems for reactive reasoning in dynamic environments.
In Schaub, T., Friedrich, G., and O'Sullivan, B., editors, *21st European Conference on Artificial Intelligence (ECAI 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 159–164. IOS Press.

Ellmauthaler, S. (2018).
Multi-Context Reasoning in Continuous Data-Flow Environments.
PhD thesis, Leipzig University.

Ellmauthaler, S. (2019).
Multi-context reasoning in continuous data-flow environments.
KI - Künstliche Intelligenz, 33(1):101–104.

References III

... a selection

Gonçalves, R., Knorr, M., and Leite, J. (2014).
Evolving multi-context systems.
In Schaub, T., Friedrich, G., and O'Sullivan, B., editors, *21st European Conference on Artificial Intelligence (ECAI 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 375–380.