Multi-Context Reasoning in Continuous Data-Flow Environments
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

[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 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 \( L = \langle KB, BS, \text{acc} \rangle \), where

- \( KB \) is a set of knowledge bases,
- \( BS \) is a set of belief sets, and
- \( \text{acc} : KB \mapsto 2^{BS} \), the *acceptance function* is a function which assigns to each knowledge base a set of belief sets.
Represent KRR Formalisms

Description Logic $\mathcal{AL}$

$L_d = \langle \text{KB}_d, \text{BS}_d, \text{acc}_d \rangle$

- $\text{KB}_d$ are all ontologies
- $\text{BS}_d$ is the set of deductively closed subsets in $\mathcal{AL}$
- $\text{acc}_d$ is a mapping of $kb \in \text{KB}_d$ to $M \subseteq 2^{\text{BS}_d}$, s.t. $\forall m \in M \; kb \models m$ holds.
Represent KRR Formalisms

Description Logic $\mathcal{AL}$

$L_d = \langle KB_d, BS_d, acc_d \rangle$

- $KB_d$ are all ontologies
- $BS_d$ is the set of deductively closed subsets in $\mathcal{AL}$
- $acc_d$ 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

$L_{asp} = \langle KB_{asp}, BS_{asp}, acc_{asp} \rangle$

- Let $A$ be the set of all possible ground atoms
- $KB_{asp}$ is the set of all answer set programs over $A$.
- $BS_{asp} = 2^A$
- $acc_{asp}$ 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

- **Context**: A context is a triple $C = \langle L, OP, mng \rangle$ where $L = \langle KB, BS, acc \rangle$ is a logic, $OP$ is a set of operations, and $mng : 2^{OP} \times KB \rightarrow KB$ is a management function.

- **Bridge Rule**: Let $C = \langle C_1, \ldots, C_n \rangle$ be a tuple of contexts and $IL = \langle IL_1, \ldots, IL_k \rangle$ a tuple of input languages. A bridge rule for $C_i$ over $C$ and $IL$, $i \in \{1, \ldots, n\}$, is of the form $op \leftarrow a_1, \ldots, a_j, \neg a_{j+1}, \ldots, \neg a_m \text{ or next } (op) \leftarrow a_1, \ldots, a_j, \neg a_{j+1}, \ldots, \neg a_m$.

- **Example**: $setTemp(hot) \leftarrow st::tmp(T), 42 < T$

Building Blocks

- **Stove Sensors**: $I_{st}$
- **Position Tracking**: $I_{pos}$
- **Medical Sensors**: $I_{ms}$
- **Drug Dispenser**: $I_{dd}$

- **Stove Storage**: $C_{st}$
- **Position Storage**: $C_{pos}$
- **Health Ontology**: $C_{ho}$
- **Health Monitor**: $C_{hm}$

- **ASP Control**: $C_{ec}$
Syntax

Definition (Context)
A context is a triple $C = \langle L, OP, mng \rangle$ where $L = \langle KB, BS, acc \rangle$ is a logic, $OP$ is a set of operations, $mng: 2^{OP} \times KB \rightarrow KB$ is a management function.

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

$$\text{op} \leftarrow a_1, \ldots, a_j, \neg a_{j+1}, \ldots, \neg a_m \text{ or next } (\text{op}) \leftarrow a_1, \ldots, a_j, \neg a_{j+1}, \ldots, \neg a_m$$

Example
$$\text{setTemp}(\text{hot}) \leftarrow \text{st}::\text{tmp}(T), 42 < T$$
$$\text{next } (\text{setPower}(\text{off})) \leftarrow \text{ec}:\text{turnOff}(\text{stove})$$
$$\text{next } (\text{setPower}(\text{off})) \leftarrow \text{st}::\text{switch}, \text{st}:\text{pw}$$
**Definition (Context)**

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

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

A context is a triple \( C = \langle L, OP, \text{mng} \rangle \) where
- \( L = \langle KB, BS, \text{acc} \rangle \) is a logic,
- \( OP \) is a set of operations,
- \( \text{mng} : 2^{OP} \times KB \rightarrow KB \) is a management function.
**Definition (Context)**

A context is a triple \( C = \langle L, OP, \text{mng} \rangle \) where

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

**Definition (Bridge Rule)**

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

\[
\text{next}(op) \leftarrow a_1, \ldots, a_j, \text{not } a_{j+1}, \ldots, \text{not } a_m
\]

or

\[
op \leftarrow a_1, \ldots, a_j, \text{not } a_{j+1}, \ldots, \text{not } a_m
\]
**Syntax**

**Definition (Context)**

A context is a triple \( C = \langle L, \text{OP}, \text{mng} \rangle \) where \( L = \langle \text{KB}, \text{BS}, \text{acc} \rangle \) is a logic, \( \text{OP} \) is a set of operations, \( \text{mng}: 2^{\text{OP}} \times \text{KB} \rightarrow \text{KB} \) is a management function.

**Example**

- \( \text{setTemp}(\text{hot}) \leftarrow \text{st::tmp}(T), 42 < T \)
- \( \text{next(setPower(}\text{off}) \leftarrow \text{ec:turnOff(stove)} \)
- \( \text{next(setPower(}\text{off}) \leftarrow \text{st::switch, st:pw} \)

**Definition (Bridge Rule)**

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

\[
\text{op} \leftarrow a_1, \ldots, a_j, \textbf{not } a_{j+1}, \ldots, \textbf{not } a_m \text{ or } \\
\text{next(op)} \leftarrow a_1, \ldots, a_j, \textbf{not } a_{j+1}, \ldots, \textbf{not } a_m
\]
Definition (Reactive Multi-Context System)

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

- $C = \langle C_1, \ldots, C_n \rangle$ is a tuple of contexts;
- $IL = \langle IL_1, \ldots, IL_k \rangle$ is a tuple of input languages;
- $BR = \langle BR_1, \ldots, BR_n \rangle$ 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

Definition (Configuration of Knowledge Bases)

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

Definition (Belief State)

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

Definition (Input)

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

S. Ellmauthaler
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

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

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

Definition (Update Function)

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

Definition (Input Stream)

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

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

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

In a dual manner, we will refer to the function $KB : [1..\tau] \rightarrow \text{Con}_M$ as the configurations stream of $M$ given $KB, I, and B$. 
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\text{(}off\text{)} \leftarrow st::switch, st:pw
- setPower\text{(}on\text{)} \leftarrow st::switch, \textbf{not } st:pw
Declarative and Operational Bridge Rules

Example

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

Declarative approach

- `setPower(off) ← st::switch, st:pw`
- `setPower(on) ← st::switch, not st: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(off) ← st::switch, st::pw
- setPower(on) ← st::switch, not st::pw
- No Equilibrium can be found

Operational approach

- next(setPower(off)) ← st::switch, st::pw
- next(setPower(on)) ← st::switch, not st::pw
Declarative and Operational Bridge Rules

Example

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

Declarative approach

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

Operational approach - without sensor data

- \text{add} (\text{switchpower}) \leftarrow \text{st::switch}
- \text{next} (\text{setPower} (\text{off})) \leftarrow \text{st:switchpower}, \text{st:pw}
- \text{next} (\text{setPower} (\text{on})) \leftarrow \text{st:switchpower}, \text{not st:pw}
Handling Time

Possible ways
- Sensor
- Time-Context

Time Context

\[
\begin{align*}
\text{setTime}(\text{now}(0)) & \leftarrow \textbf{not} \text{ clock:timeAvailable} \\
\text{next}(\text{add}(\text{timeAvailable})) & \leftarrow \text{clock:now}(0) \\
\text{next}(\text{setTime}(\text{now}(T + 1))) & \leftarrow \text{clock:now}(T)
\end{align*}
\]
Forgetting and Windowing

Volatile Information and Reasoning with a Window

\[
\text{next}(\text{add}(\text{alert}(\text{stove}, T))) \leftarrow c::\text{now}(T), ec:\text{alert}(\text{stove}).
\]
\[
\text{next}(\text{del}(\text{alert}(\text{stove}, T))) \leftarrow stE:\text{alert}(\text{stove}, T), \textbf{not} ec:\text{alert}(\text{stove}).
\]
\[
\text{add}(\text{emergency}(\text{stove})) \leftarrow c::\text{now}(T), ec:\text{alert}(\text{stove}),
\quad stE:\text{alert}(\text{stove}, T'),
\quad stE:\text{winE}(Y), |T - T'| \geq Y.
\]

Dynamic Window

\[
\text{next}(\text{set}(\text{win}(P, X))) \leftarrow ed:\text{defWin}(P, X), \textbf{not} ed:\text{ susp}(E).
\]
\[
\text{next}(\text{set}(\text{win}(P, Y))) \leftarrow ed:\text{rel}(P, E, Y), ed:\text{ susp}(E).
\]
\[
\text{alarm}(E) \leftarrow ed:\text{conf}(E).
\]
\[
\text{next}(\text{add}(P(T))) \leftarrow c::\text{now}(T), s::P.
\]
\[
\text{next}(\text{del}(P(T'))) \leftarrow ed:P(T'), c::\text{now}(T), ed:\text{win}(P, Z), T' < T - Z.
\]
Thank you for your interest
Equilibria in heterogeneous nonmonotonic multi-context systems.

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.

Reactive multi-context systems: Heterogeneous reasoning in dynamic environments.
Artificial Intelligence, 256:68–104.

