Asynchronous Multi-Context Systems¹

Stefan Ellmauthaler  Jörg Pührer

Computer Science Institute
Leipzig University
Germany

ReactKnow 2014
Prague
August 19th 2014

¹Research has been supported by DFG and FWF (projects BR 1817/7-1 and FOR 1513)
Outline

1 Motivation

2 Concept

3 Asynchronous Multi-Context Systems

4 Relation to (reactive) Multi-Context Systems

5 Outlook & Related Work
Motivation

- **integration** of heterogeneous KR-formalisms
- **awareness** of continuous flow of knowledge
reactive Multi-Context Systems so far ...

**Motivation**
- **integration** of heterogeneous KR-formalisms
- **awareness** of continuous flow of knowledge

**Realisation**
reactive Multi-Context Systems so far ...

Motivation
- **integration** of heterogeneous KR-formalisms
- **awareness** of continuous flow of knowledge

Realisation
- **Contexts** with different KR & Reasoning formalisms
- **Bridge-Rules** for exchange of beliefs
- **Notion of Equilibrium** as Semantics
- **Run** represents the change of knowledge and belief over time
... and a slight look to online-applications

- Many different services and sources of knowledge
- Continuous flow of information
- Data collection till sufficient knowledge for their tasks is available
- Communication is often query-based
- Asynchronous communication protocols
... and a slight look to online-applications

- Data collection till sufficient knowledge for their tasks is available
- Communication is often query-based
- Asynchronous communication protocols
Example Environment - Emergency Team Management

- Emergency Call
- Classification and Prioritisation of each case
- Overview of available rescue units
- Overview on ETAs for each unit and case
- Suggesting optimal assignments
- Communicate Tasks to rescue units
Computer Aided Emergency Team Management

Example Environment - Emergency Team Management

- Emergency Call
- Classification and Prioritisation of each case
- Overview of available rescue units
- Overview on ETAs for each unit and case
- Suggesting optimal assignments
- Communicate Tasks to rescue units

Requirements

- Fast response to events
- Consider different sources of data
- Modularity for additional components
- Human as last instance for decisions
Consequences of Asynchronicity

→ Contexts compute their belief sets independently
Consequences of Asynchronicity

→ Contexts compute their belief sets independently
→ No agreement on a common Equilibrium
Consequences of Asynchronicity

→ Contexts compute their belief sets independently
→ No agreement on a common Equilibrium
→ No defined basis for Bridge-Rules to fire
Consequences of Asynchronicity

→ Contexts compute their belief sets independently
→ No agreement on a common Equilibrium
→ No defined basis for Bridge-Rules to fire
→ Need for Output-Rules (OR)
Consequences of Asynchronicity

→ Contexts compute their belief sets independently
  → No agreement on a common Equilibrium
    → No defined basis for Bridge-Rules to fire
      → Need for Output-Rules (OR)
        → Keep track of information provided by OR
Consequences of Asynchronicity

→ Contexts compute their belief sets independently
→ No agreement on a common Equilibrium
→ No defined basis for Bridge-Rules to fire
→ Need for Output-Rules (OR)
→ Keep track of information provided by OR
→ Input stream for each context
Consequences of Asynchronicity

→ Contexts compute their belief sets independently
→ No agreement on a common Equilibrium
→ No defined basis for Bridge-Rules to fire
→ Need for Output-Rules (OR)
→ Keep track of information provided by OR
→ Input stream for each context
→ Interaction with environment:
  ▶ aMCS wide input streams
  ▶ aMCS wide output streams
Other Design-Choices

- Each context decide when to compute
  - realised by computation controller $cc$
- Dynamic adjustments of context-management
  - computation controller ($cc$)
  - output rules ($OR$)
  - context-semantics ($ACC$)
  - context update function ($cu$)
Asynchronous Multi-Context Systems

**Definition**

A *data package* is a pair $d = \langle s, I \rangle$, where $s \in \mathcal{N}$ is either a context name or a sensor name, stating the *source* of $d$, and $I \subseteq \mathcal{IL}$ is a set of pieces of information. An *information buffer* is a sequence of data packages.
Asynchronous Multi-Context Systems

**Definition**

A *data package* is a pair $d = \langle s, I \rangle$, where $s \in \mathcal{N}$ is either a context name or a sensor name, stating the *source* of $d$, and $I \subseteq \mathcal{IL}$ is a set of pieces of information. An *information buffer* is a sequence of data packages.

**Definition**

Let $C = \langle n, \mathcal{LS} \rangle$ be a context. An *output rule* $r$ for $C$ is an expression of the form

$$\langle n, i \rangle \leftarrow b_1, \ldots, b_j, \text{not } b_{j+1}, \ldots, \text{not } b_m,$$

(1)

such that $n \in \mathcal{N}$ is the name of a context or an output stream, $i \in \mathcal{IL}$ is a piece of information, and every $b_\ell$ ($1 \leq \ell \leq m$) is a belief for $C$, i.e., $b_\ell \in S$ for some $S \in \mathcal{BS}_{\mathcal{LS}}$. 
Asynchronous Multi-Context Systems

Definition

Let $C = \langle n, \mathcal{L}S \rangle$ be a context, $\text{OR}$ a set of output rules for $C$, $S \in \mathcal{BS}_{\mathcal{L}S}$ a belief set, and $n' \in \mathcal{N}$ a name. Then, the data package

$$d_C(S, \text{OR}, n') = \langle n, \{ i \mid r \in \text{OR}, \text{hd}(r) = \langle n', i \rangle, S \models \text{bd}(r) \} \rangle$$

is the output of $C$ with respect to $\text{OR}$ under $S$ relevant for $n$. 

S. Ellmauthaler, J. Pührer
ReactKnow 2014 - Prague
CSI Leipzig 9 / 19
Asynchronous Multi-Context Systems

Definition

Let $C = \langle n, \mathcal{LS} \rangle$ be a context. A configuration of $C$ is a tuple $cf = \langle KB, ACC, ib, cm \rangle$, where $KB \in \mathcal{KB}_{\mathcal{LS}}$, $ACC \in \mathcal{ACC}_{\mathcal{LS}}$, $ib$ is a finite information buffer, and $cm$ is a context management for $C$ which is a triple $cm = \langle cc, cu, OR \rangle$, where

- $cc$ is a computation controller for $C$,
- $OR$ is a set of output rules for $C$, and
- $cu$ is a context update function for $C$ which is a function that maps an information buffer $ib = d_1, \ldots, d_m$ and an admissible knowledge base of $\mathcal{LS}$ to a configuration $cf' = \langle KB', ACC', ib', cm' \rangle$ of $C$ with $ib' = d_k, \ldots, d_m$ for some $k \geq 1$. 
Asynchronous Multi-Context Systems

\[ s_1 \rightarrow C_1 \rightarrow s_2 \rightarrow C_2 \rightarrow s_3 \rightarrow C_3 \rightarrow aMCS_M \]

S. Ellmauthaler, J. Pührer
ReactKnow 2014 - Prague
CSI Leipzig 11 / 19
Run of an aMCS

Configuration of an aMCS

- Configuration for each Context
- Content of each output stream (output buffer)
Run of an aMCS

**Configuration of an aMCS**

- Configuration for each Context
- Content of each output stream (output buffer)

**Definition (Run structure)**

Let $M = \langle \langle C_1, \ldots, C_n \rangle, \langle o_1, \ldots, o_m \rangle \rangle$ be an aMCS. A run structure for $M$ is a sequence

$$R = \ldots, Cf^t, Cf^{t+1}, Cf^{t+2}, \ldots,$$

where $t \in \mathbb{Z}$ is a point in time, and every $Cf^{t'}$ in $R$ ($t' \in \mathbb{Z}$) is a configuration of $M$. 
Run of an aMCS

Time-awareness
- Computation of belief sets takes time
- Enumeration of belief sets takes time
- Verification of non-existence of (further) belief sets takes time

Run execution
- If a Context finds a belief set, OR are applied
- Information is distributed to input-buffers of contexts or output streams
- If a Context has finished its computation, EOC is sent to all stakeholders
Example of an aMCS
Differences to rMCS

- rMCSs use equilibria
  - strong semantics
  - tight integration approach where context semantics are interdependent
  - every context need to agree → each context needs to wait
Differences to rMCS

- rMCSs use equilibria
  - strong semantics
  - tight integration approach where context semantics are interdependent
  - every context need to agree → synchronous approach
Differences to rMCS

- rMCSs use equilibria
  - strong semantics
  - tight integration approach where context semantics are interdependent
  - every context need to agree → synchronous approach
- rMCSs have equilibria as source of non-determinism
- aMCSs have computation time as source of non-determinism
Simulation of rMCS

For each Context $C_i$ of the rMCS, introduce three aMCS Contexts:

- $C_i^{kb}$ stores its current knowledge base
- $C_i^{kb'}$ stores update of the knowledge base and compute its semantics
- $C_i^m$ implements the bridge rules and the management function
Simulation of rMCS

For each Context $C_i$ of the rMCS, introduce three aMCS Contexts:

- $C_{kb}^i$ stores its current knowledge base
- $C_{kb'}^i$ stores update of the knowledge base and compute its semantics
- $C_m^i$ implements the bridge rules and the management function

Three contexts for the rMCS, where

- $C^{obs}$ receives sensor data and distributes the information,
- $C^{guess}$ guesses equilibrium candidates and propagates them to $C_m^i$, and
- $C^{check}$ compares all results of the contexts and informs other contexts if an equilibrium has been found
Future Work

- Study modelling patterns and best practices for aMCS
- Analysis how other approaches (e.g., clingo [Gebser et al., 2012]) are capable for prototypical implementations
- Analysis how features of other approaches (e.g., online queries, iterative computing, ...) relate to aMCS concepts (e.g., ib, OR, cc, ...)

S. Ellmuthaler, J. Pührer
ReactKnow 2014 - Prague
CSI Leipzig 17 / 19
Related Work

- aMCS are motivated by rMCS [Brewka et al., 2014] and are based on MCS [Brewka et al., 2011a, Brewka et al., 2011b]
- Evolving Mulit-Context Systems [Gonçalves et al., 2014] follow a similar approach to rMCS
- A concept similar for output rules in reactive Multi-Context Systems has been presented as reactive bridge rules [Ellmauthaler, 2013]
Thank you for your interest!
[Brewka et al., 2011a] Brewka, G., Eiter, T., and Fink, M. (2011a).
Nonmonotonic multi-context systems: A flexible approach for integrating heterogeneous knowledge sources.
In Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning, pages 233–258. Springer.

[Brewka et al., 2011b] Brewka, G., Eiter, T., Fink, M., and Weinzierl, A. (2011b).
Managed multi-context systems.
In IJCAI’11, pages 786–791.

[Brewka et al., 2014] Brewka, G., Ellmauthaler, S., and Pührer, J. (2014).
Multi-context systems for reactive reasoning in dynamic environments.
In Proc. ECAI’14.
To appear.
[Ellmauthaler, 2013] Ellmauthaler, S. (2013).
Generalizing multi-context systems for reactive stream reasoning applications.
In Proc. ICCSW’13, pages 17–24.

[Gebser et al., 2012] Gebser, M., Grote, T., Kaminski, R., Obermeier, P., Sabuncu, O., and Schaub, T. (2012).
Stream reasoning with answer set programming: Preliminary report.
In Proc. KR’12.

[Gonçalves et al., 2014] Gonçalves, R., Knorr, M., and Leite, J. (2014).
Evolving multi-context systems.
In Proc. ECAI’14.
To appear.
Definition

A context is a pair $C = \langle n, \mathcal{LS} \rangle$ where $n \in \mathcal{N}$ is the name of the context and $\mathcal{LS}$ is a logic suite.
Definition

An aMCS (of length $n$ with $m$ output streams) is a pair $M = \langle C, O \rangle$, where $C = \langle C_1, \ldots, C_n \rangle$ is an $n$-tuple of contexts and $O = \langle o_1, \ldots, o_m \rangle$ with $o_j \in \mathcal{N}$ for each $1 \leq j \leq m$ is a tuple containing the names of the output streams of $M$. 
Definition

A data package is a pair $d = \langle s, I \rangle$, where $s \in \mathcal{N}$ is either a context name or a sensor name, stating the source of $d$, and $I \subseteq \mathcal{I} \mathcal{L}$ is a set of pieces of information. An information buffer is a sequence of data packages.
Definition

Let $C = \langle n, \mathcal{LS} \rangle$ be a context. A computation controller for $C$ is a relation $cc$ between a KB $KB \in \mathcal{KB}_{\mathcal{LS}}$ and a finite information buffer.
Definition

Let $C = \langle n, \mathcal{L}S \rangle$ be a context. An output rule $r$ for $C$ is an expression of the form

$$\langle n, i \rangle \leftarrow b_1, \ldots, b_j, \text{not } b_{j+1}, \ldots, \text{not } b_m,$$

such that $n \in \mathcal{N}$ is the name of a context or an output stream, $i \in \mathcal{IL}$ is a piece of information, and every $b_{\ell}$ $(1 \leq \ell \leq m)$ is a belief for $C$, i.e., $b_{\ell} \in S$ for some $S \in \mathcal{BS}_{\mathcal{L}S}$. 

(2)
Definition

Let $C = \langle n, \mathcal{LS} \rangle$ be a context, OR a set of output rules for $C$, $S \in \mathcal{BS}_{\mathcal{LS}}$ a belief set, and $n' \in \mathcal{N}$ a name. Then, the data package

$$d_C(S, OR, n') = \langle n, \{ i \mid r \in OR, hd(r) = \langle n', i \rangle, S \models bd(r) \} \rangle$$

is the output of $C$ with respect to OR under $S$ relevant for $n$. 
Definition

Let $C = \langle n, \mathcal{LS} \rangle$ be a context. A configuration of $C$ is a tuple $cf = \langle KB, ACC, ib, cm \rangle$, where $KB \in KB_{\mathcal{LS}}$, $ACC \in ACC_{\mathcal{LS}}$, $ib$ is a finite information buffer, and $cm$ is a context management for $C$ which is a triple $cm = \langle cc, cu, OR \rangle$, where

- $cc$ is a computation controller for $C$,
- $OR$ is a set of output rules for $C$, and
- $cu$ is a context update function for $C$ which is a function that maps an information buffer $ib = d_1, \ldots, d_m$ and an admissible knowledge base of $\mathcal{LS}$ to a configuration $cf' = \langle KB', ACC', ib', cm' \rangle$ of $C$ with $ib' = d_k, \ldots, d_m$ for some $k \geq 1$. 
Definition

Let $M = \langle\langle C_1, \ldots, C_n\rangle, \langle o_1, \ldots, o_m\rangle\rangle$ be an aMCS. A configuration of $M$ is a pair

$$Cf = \langle\langle cf_1, \ldots, cf_n\rangle, \langle ob_1, \ldots, ob_m\rangle\rangle,$$

where

- for all $1 \leq i \leq n$ $cf_i = \langle KB, ACC, ib, cm\rangle$ is a configuration for $C_i$ and for every output rule $r \in OR_{cm}$ we have $n \in N(M)$ for $\langle n, i\rangle = hd(r)$, and
- $ob_j = \ldots, d_{l-1}, d_l$ is an information buffer with a final element $d_l$ that corresponds to the data on the output stream named $o_j$ for each $1 \leq j \leq m$ such that for each $h \leq l$ with $d_h = \langle n, i\rangle$ we have $n = n_{C_i}$ for some $1 \leq i \leq n$. 

S. Ellmauthaler, J. Pührer  
ReactKnow 2014 - Prague  
CSI Leipzig 9 / 12
Definition

Let $M = \langle \langle C_1, \ldots, C_n \rangle, \langle o_1, \ldots, o_m \rangle \rangle$ be an aMCS. A run structure for $M$ is a sequence

$$R = \ldots, Cf^t, Cf^{t+1}, Cf^{t+2}, \ldots,$$

where $t \in \mathbb{Z}$ is a point in time, and every $Cf^{t'}$ in $R$ ($t' \in \mathbb{Z}$) is a configuration of $M$. 
Definition

Let $M$ be an aMCS of length $n$ with $m$ output streams and $R$ a run structure for $M$. $R$ is a run for $M$ if the following conditions hold for every $1 \leq i \leq n$ and every $1 \leq j \leq m$:

(i) if $cf_i^t$ and $cf_i^{t+1}$ are defined, $C_i$ is neither busy nor waiting at time $t$, then

- $C_i$ is busy at time $t + 1$,
- $cf_i^{t+1} = cu_{cm_i}(ib_i^t, KB_i^t)$

We say that $C_i$ started a computation for $KB_i^{t+1}$ at time $t + 1$.

(ii) if $C_i$ started a computation for $KB$ at time $t$ then

- we say that this computation ended at time $t'$, if $t'$ is the earliest time point with $t' \geq t$ such that $\langle n_{C_i}, EOC \rangle$ is added to every stakeholder buffer $b$ of $C_i$ at $t'$; the addition of $d_{C_i}(S, OR_{cm_i''}, n)$ to $b$ is called an end of computation notification.
- for all $t' > t$ such that $cf_i^{t'}$ is defined, $C_i$ is busy at $t'$ unless the computation ended at some time $t''$ with $t < t'' < t'$.
- if the computation ended at time $t'$ and $cf_i^{t'+1}$ is defined then $C_i$ is not busy at $t' + 1$. 
(iii) if \( C_i \) started a computation for KB at time \( t \) that ended at time \( t' \) then for every belief set \( S \in \text{ACC}^t_i \) there is some time \( t'' \) with \( t \leq t'' \leq t' \) such that

\[ d_{C_i}(S, \text{OR}_{cm_i}^{t''}, n) \text{ is added to every stakeholder buffer } b \text{ of } C_i \text{ for } n \text{ at } t''. \]

We say that \( C_i \) computed \( S \) at time \( t'' \). The addition of \( d_{C_i}(S, \text{OR}_{cm_i}^{t''}, n) \) to \( b \) is called a belief set notification.

(iv) if \( \text{ob}_j^t \) and \( \text{ob}_j^{t+1} \) are defined and \( \text{ob}_j^t = \ldots, d_{l-1}, d_l \) then \( \text{ob}_j^{t+1} = \ldots, d_{l-1}, d_l, \ldots, d_{l'} \) for some \( l' \geq l \). Moreover, every data package \( d_{l''} \) with \( l < l'' \leq l' \) that was added at time \( t + 1 \) results from an end of computation notification or a belief set notification.

(v) if \( \text{cf}_i^t \) and \( \text{cf}_i^{t+1} \) are defined, \( C_i \) is busy or waiting at time \( t \), and \( \text{ib}_i^t = d_1, \ldots, d_l \) then we have \( \text{ib}_i^{t+1} = d_1, \ldots, d_l, \ldots, d_{l'} \) for some \( l' \geq l \). Moreover, every data package \( d_{l''} \) with \( l < l'' \leq l' \) that was added at time \( t + 1 \) either results from an end of computation notification or a belief set notification or \( n \notin \mathcal{N}(M) \) (i.e., \( n \) is a sensor name) for \( d_{l''} = \langle n, i \rangle \).