Stream Reasoning and Multi-Context Systems

Thomas Eiter

Institute of Logic and Computation
Vienna University of Technology (TU Wien)

joint work with M. Dao-Tran[1], A. Falkner[3], P. Ogris[2], K. Schekotihin[2],
P. Schneider[1,3], P. Schüller[1], A. Weinzierl[1]

(1) (2) (3)

Stream Reasoning Workshop 2019,
Linköping, Sweden, April 16-17, 2019

- Austrian Science Fund (FWF) grant P26471
- Austrian Research Promotion Agency (FFG), 588655

Outline

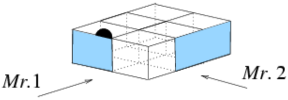
Multi-Context Systems

- **Contextual Reasoning:** model information interlinkage of knowledge bases / agents
  - information flow between KBs via *bridge rules*

$$
\begin{aligned}
Mr.1 : \quad & row(X) \leftarrow (Mr.2 : sees\_row(X)) \\
Mr.2 : \quad & col(Y) \leftarrow (Mr.1 : sees\_col(Y))
\end{aligned}
$$

  - equilibrium ensures aligned information

Ghidina & Giunchiglia's Magic Box



- Different early varieties
  - Trento School (Giunchiglia, Serafini et al.):
    - Heterogeneous MCS [Giunchiglia and Serafini, 1994]
    - Nonmonotonic bridge rules [Roelofsen and Serafini, 2005]
    - Extension to Contextual Default Logic [Brewka *et al.*, 2007]
  - *nonmonotonic multi-context systems (MCS)* [Brewka and E_, 2007]
  - *managed MCS (mMCS)* [Brewka *et al.*, 2011]

Nonmonotonic Multi-Context Systems (MCS)

Multi-Context System

Formally, a Multi-Context System

$$M = (C_1, \ldots, C_n)$$

consists of contexts

$$C_i = (L_i, kb_i, br_i),\ i \in \{1, \ldots, n\},$$

where

- each L_i is a "logic,"
- each kb_i is a knowledge base in L_i, and
- each br_i is a set of L_i-bridge rules over M's logics.

Logic

A *logic* L is a tuple $L = (\mathbf{KB}_L, \mathbf{BS}_L, \mathbf{ACC}_L)$, where

- $\mathbf{KB}_L$ is a set of well-formed knowledge bases, each being a set (of "formulas")
- $\mathbf{BS}_L$ is a set of possible belief sets, each being a set (of "formulas")
- $\mathbf{ACC}_L : \mathbf{KB}_L \rightarrow 2^{\mathbf{BS}_L}$ assigns each KB a set of acceptable belief sets

Thus, logic L caters for multiple extensions of a knowledge base.

Bridge Rules

A L_i-bridge rule over logics $L_1, \ldots, L_n$, $1 \leq i \leq n$, is of the form

$$s \leftarrow (r_1 : p_1), \ldots, (r_j : p_j), not(r_{j+1} : p_{j+1}), \ldots, not(r_m : p_m)$$

where $kb \cup \{s\} \in \mathbf{KB}_i$ for each $kb \in \mathbf{KB}_i$, each $r_k \in \{1, \ldots, n\}$, and each p_k is in some belief set of L_{r_k}.

Note: such rules are akin to rules of normal logic programs

Example (Authors)

Suppose a MCS $M = (C_1, C_2)$ has contexts that express the individual views of a paper by the two authors.

- C_1:
  - L_1 = Classical Logic
  - $kb_1 = \{ unhappy \supset revision \}$
  - $br_1 = \{ unhappy \leftarrow (2 : work) \}$
- C_2:
  - L_2 = Reiter's Default Logic
  - $kb_2 = \{ good : accepted / accepted \}$
  - $br_2 = \{ work \leftarrow (1 : revision),$
    $good \leftarrow not(1 : unhappy) \}$

Equilibrium Semantics

- **Belief State**

  A *belief state* is a sequence $S = (S_1, \ldots, S_n)$ of belief sets S_i in L_i

- **Applicable Bridge Rules**

  For $M = (C_1, \ldots, C_n)$ and belief state $S = (S_1, \ldots, S_n)$, the bridge rule

  $$s \leftarrow (r_1 : p_1), \ldots, (r_j : p_j), not(r_{j+1} : p_{j+1}), \ldots, not(r_m : p_m)$$

  is *applicable in* S if (1) $p_i \in S_{r_i}$, for $1 \leq i \leq j$, and (2) $p_k \notin S_{r_k}$, for $j < k \leq m$.

- **Equilibrium**

  A belief state $S = (S_1, \ldots, S_n)$ of M is an equilibrium iff for all $i = 1, \ldots, n$,

  $$S_i \in \mathbf{ACC}_i(kb_i \cup \{head(r) \mid r \in br_i \text{ is applicable in } S\})\ .$$

Equilibrium Semantics, cont'd

Example, cont'd

Reconsider $M = (C_1, C_2)$:

- $kb_1 = \{\, unhappy \supset revision \,\}$ (Classical Logic)
- $br_1 = \{\ unhappy \leftarrow (2 : work)\ \}$

- $kb_2 = \{\, good : accepted/accepted \,\}$ (Default Logic)
- $br_2 = \{\ work \leftarrow (1 : revision),$
  $good \leftarrow not(1 : unhappy)\ \}$

M has two equilibria:

- $E_1 = (Th(\{unhappy, revision\}), Th(\{work\}))$ and
- $E_2 = (Th(\{unhappy \supset revision\}), Th(\{good, accepted\}))$

Managed MCS

- MCS: pure information alignment, fully static
- introduce *context manager*, to *update/change* the KB
  - Bridge rules:

    $$op(f) \leftarrow (c_1\!:\!p_1), \ldots, (c_j\!:\!p_j), not(c_{j+1}\!:\!p_{j+1}), \ldots, not(c_m\!:\!p_m).$$
  - *management function* $mng : 2^{F_{LS}^{OP}} \times KB_{LS} \rightarrow 2^{(KB_{LS} \times \mathcal{ACC}_{LS})} \setminus \{\emptyset\}$

    assigns updates commands + KB a follow-up KB + evaluation semantics

- *managed context* $C_i = (LS_i, kb_i, br_i, OP_i, mng_i)$ with
  - $LS_i = (BS_{LS_i}, KB_{LS_i}, \mathcal{ACC}_{LS_i})$ a logic suite,
  - $kb_i \in KB_{LS_i}$ a knowledge base,
  - br_i a set of bridge rules for C_i,
  - OP_i a management base (commands), and
  - mng_i a management function over LS_i and OP_i.

- **Managed Multi-Context System (mMCS)** $M = (C_1, \ldots, C_n)$ are stateful, form the basis of other MCS (eMCs, rMCS, aMCS, sMCS, dMCS, tMCS)

Managed MCS, cont'd

Example (Diseases)

- C_1: relational database on disease treatments

$$kb_1 = \{\, treat(pen, str\_pneu, pneu, evd), treat(azith, leg\_pneu, leg, evd), ineff(pen, leg\_pneu)\}$$

conclude likely effects using C_2.

$$br_1 = \{\, treat(X, B, I, likely) \leftarrow (1 : treat\ (X, B, \_, \_))\ ,(2 : B\ rdf{:}causes\ I)\ .\}.$$

- C_2: RDF-triple store on disease causations.

$$kb_2 = \{str\_pneu\ rdf{:}causes\ men, leg\_pneu\ rdf{:}causes\ atyp\_pneu\}.$$

- C_3: bacteria ontology (DL)
- C_4: generalized logic program deriving possible medication effects:

$$\begin{aligned} br_4 = \{\ & add(isa(X,Y)) \leftarrow (3 : (X \sqsubseteq Y))\ . \\ & add(eff(X,B)) \leftarrow (1 : eff\ (X,B))\ . \\ & upd(\text{not}\ eff(X,B)) \leftarrow (1 : ineff\ (X,B))\ .\}, \end{aligned}$$

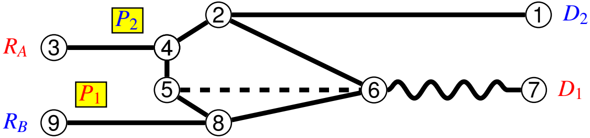
Semantics

- Applicable bridge rule heads: $app_i(S) = \{hd(r) \mid r \in br_i \land S \models body(r)\}$.
- Equilibrium: $S = (S_1, \ldots, S_n)$ iff for every $1 \leq i \leq n$ some $(kb'_i, ACC_{LS_i}) \in mng_i(app_i(S), kb_i)$ exists s.t. $S_i \in ACC_{LS_i}(kb'_i)$.

Streaming World

- Sensors, networks, mobile devices:
  - getting to a connected world...
- Pushing rather than pulling of data
- Dynamic streams of data, potentially infinite
  - low frequency changes (meter reading)
  - high frequency changes (stock trading)
- Continuous computation / evaluation
  - synchronous vs. asynchronous
- Reference to time
- **Poses challenges to MCSs**

Example: Cooperative Robots



- In a mall, robots must deliver packages to destinations
- R_A must deliver package P_1 (at 9) to destination D_1 (7)
- R_B must deliver package P_2 (at 4) to destination D_2 (1)
- Minimize travel distance: agree to pick up other package and exchange (e.g. at node 5)
- Agreement may be challenging: robots already move, connections turn out unusable (too many people around), ...
- **Setting**: *dynamic monitoring* of usability

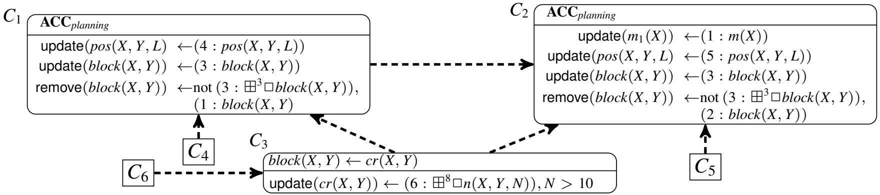
  sensors for position, occupation etc.

MCS Features

- (static) **MCS, mMCS**: have an equilibrium (fixpoint) semantics
- (dynamic) **reactive MCS (rMCS)** [Brewka et al., 2014,2018],
  evolving MCS (eMCS) [Gonçalves *et al.*, 2014]:
  - computing equilibria is timeless
- (dynamic) **asynchronous MCS (aMCS)** [Ellmauthaler and Pührer, 2015]:
  - physical computation time, transfer time are disregarded
  - no baseline mechanism to achieve equilibrium
- **streaming MCS (sMCS)** [Dao-Tran and E_, 2017]:
  - bridge rules with *window atoms* (simple LARS formulas [Beck *et al.*, 2018]) to access input streams
  - model *computation time* and *data transfer time*
  - internal *asynchronous execution control* (restart/wait on eval requests)
  - run-based semantics, with *feedback equilibria* to enforce *local stability* in runs (avoid infinite loops, and generalize rMCS, eMCS)

additional stream reasoning inside contexts possible!

sMCS by Example (cont'd)



- sensor context C_i, $4 \leq i \leq 6$ feeds sensor input to C_{i-3}
- C_4 (C_5) tells position $pos(X,Y,L)$ of R_A (R_B) on $X \to Y$, $L \in \{0\%, \ldots, 100\%\}$
- C_3 gets sensor data of C_6, infers blocked connections and sends this info to C_1, C_2
- C_1 (C_2): shortest route for R_A (R_B) to D_1 (D_2), with blocked connections, meeting point $m(X)$

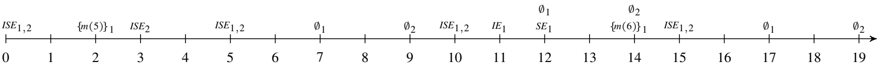
bridge rules: *window atoms* $(6\colon ⊞^8\ \Box n(X,Y,N))$, $(3\colon ⊞^3\ \Box block(X,Y))$

- update$(cr(X,Y)) \leftarrow (6 : ⊞^8 \Box n(X,Y,N)), N > 10$ accesses C_6's output
  - "store link $X \to Y$ is crowded, if C_6 reported in the last 8 mins always 10 people on it."
- remove$(block(X,Y)) \leftarrow$ not $(3 : ⊞^3 \Box block(X,Y)), (2 : block(X,Y))$
  - "unblock link $X \to Y$ unblocked, if C_3 didn't report it the last 3 mins always blocked."

Run-based Semantics

- A *state* of C_i is a triple $\mathbf{s}_i{=}(s_i, o_i, kb_i)$ where
  - $s_i \subseteq \{IE, SE\}$ is the *execution status* (intend to execute / start to execute)
  - $o_i \subseteq Bel_i$ is the *output belief set* streamed to other contexts, unless $o_i = \epsilon$;
  - kb_i is the local KB (which can evolve).
- *runs* are constrained state sequences $\mathbf{s} = \mathbf{s}(0), \ldots, \mathbf{s}(t)$ of *global states* $\mathbf{s}(t') = (\mathbf{s}_1(t'), \ldots, \mathbf{s}_n(t'))$, where each $\mathbf{s}_i(t')$ is state of C_i:
  - delay intention IE to actual start SE (busy) or restart
  - respect data transfer time Δ_{ki}, computation time $f_i(br_i, kb_i)$

Example (run trace) (focus on C_1, C_2; $ISE_{1,2} = \{IE_1, SE_1, IE_2, SE_2\}$, $ISE_2 = \{IE_2, SE_2\}$)



- sensors C_4, C_5 stream at $5k \geq 0$, C_6 streams continuously
- C_1, C_2, C_3 run in pushing mode; C_1 ignores new input when busy; C_2, C_3 restart.
- $\Delta_{12} = \Delta_{31} = 1$ and $\Delta_{ij} = 0$ otherwise
- $f_1(br_1, kb_1){=}2$, $f_2(br_2, kb_2){=}4$, and $f_3(br_3, kb_3){=}1$ for all kb_i
- C_3 sends $block(5, 6)$ at $t = 10$ (ignored by C_1 at 11)

Idealized Runs

- **Aim:** capture rMCS and eMCS which feature step-wise equilibrium computation at zero cost (= infinite speed)
- **Naive Approach:** set in runs $\Delta_{ki} = f_i = 0$
  - **does not work:** local KBs kb_i could not change (but rMCS/eMCS runs are stateful)
- **Way out:** extend time ontology with *infinitesimally small chronon* ε
  - computation time is ε
  - $t < t + \varepsilon$ and $t + \epsilon = t + k\varepsilon$ for each $k \in \mathbb{N} > 0$
  - transfer time $f_i = 0$
- In an *idealized run* $\mathbf{s} = (\mathbf{s}_0, \ldots, \mathbf{s}_{t_{end}})$,

  $$o_i(t{+}1) = o_i(t{+}\varepsilon) \text{ and } kb_i(t{+}1) = kb_i(t{+}\varepsilon),$$

  where
  - $o_i(t+\varepsilon) = \mathbf{ACC}(kb_i(t+\varepsilon))$ and
  - $kb_i(t+\varepsilon) = mng_i(app_i^{\varepsilon}(\mathbf{s}, t), kb_i(t))$

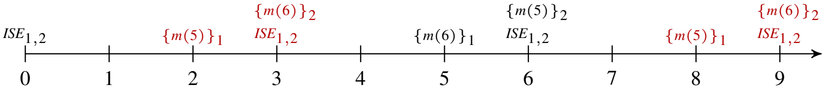
Emulation Property: runs of an rMCS M (resp. eMCS M from a natural class) can be emulated by idealized runs of a corresponding sMCS M^s

Feedback Equilibria

- Total asynchrony can be uncomfortable if contexts depend on each other
- Suppose also C_2 suggests a meeting point, imported by C_1 into kb_1 via

$$\text{update}(m_2(X)) \leftarrow (2{:}m(X))$$

- Meeting mismatch: do a further round (e.g., follow other proposal)
- Assume $f_1(br_1, kb_1){=}2, f_2(br_2, kb_2) = 3$ for all kb_i , $\Delta_{12} = 1$, other costs=0:



computing an agreed meeting point loops indefinitely

Feedback Equilibria, cont'd

- **Key ideas:**
  - consider *strongly connected components (SCCs)* via an import graph ($C_i \rightarrow C_j$, if $(j{:}A)$ occurs in br_i)
  - for equilibrium computation, *dispense streaming data from outside*
  - any C_i can request, while computing, at time t stability of its SCC $\mathcal{C}_i$:
    - the contexts in $\mathcal{C}_i$ are restarted with input at t_{exe}
    - at some $t'' \geq t$, either $\mathcal{C}_i$ reports an equilibrium, or $\mathcal{C}_i$ restarts its contexts with input at time t'' if no equilibrium exists

- **Feedback Equilibrium** of $\mathcal{C}_i$ wrt. run $\mathbf{s}$ at time t: for every $C_{i_j} \in \mathcal{C}_i$,

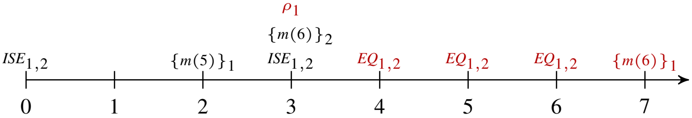
$$\text{belief set } BS_{i_j} \in \mathbf{ACC}_{i_j}(mng_{i_j}(app^{\varepsilon}_{i_j}(\mathbf{s}, t), kb_{i_j}(t)))$$

  - intuitively, *run in idealized mode*
  - *input* to $\mathcal{C}_i$ *is frozen* at t
  - inside $\mathcal{C}_i$, *cyclic information flow* is respected

Locally Stable Runs

- ρ denotes a stability request, EQ is a new status
- Informally a *locally stable run* for $\mathcal{C}$ is a run $\mathbf{s} = (\mathbf{s}_0, \ldots, \mathbf{s}_{t_{end}})$, s.t.
  - requests for local stability of $\mathcal{C}$ at time t are granted at $t' \geq t$;
  - to serve a request, all $C \in \mathcal{C}_i$ switch to equilibrium computation from t;
  - a feedback eq. is returned at output time t_{out} if one exists, else $\mathcal{C}$ is restarted.
- A run $\mathbf{s}$ is *locally stable*, if it is locally stable for each SCC

Example



- C_1 realizes at $t = 3$ that C_2's suggestion does not match his of $t = 2$.
- C_1 requests local stabilization for $\mathcal{C}_1 = \{C_1, C_2\}$
- meeting at node 6 yields an equilibrium at time 7

Reasoning

- **Setting:** sMCS $M=(C_1,\ldots,C_n)$
  - sensor contexts $O=C_j, C_{j+1},\ldots,C_n$
  - reading $\mathbf{r}=\mathbf{r}(0),\ldots,\mathbf{r}(t)$: sensor data stream
  - periodic decision to execute, permanent ignore/restart
  - algs to evaluate br_i, mng_i, $\mathbf{ACC}_i$ (compute some $BS_i \in \mathbf{ACC}_i(kb_i)$)
- **Monitoring:** watch the past
  - $M,\mathbf{r} \models_b C_i(a)$: a is believed at C_i at time t in some run $\mathbf{s} = (\mathbf{s}_0,\ldots,\mathbf{s}_t)$ for $\mathbf{r}$
  - $M,\mathbf{r} \models_{bs} C_i(a)$: ... some locally stable run ...
- **Prediction:** consider future data
  - $M,\mathbf{r} \models_x^{t'} C_i(a)$: does $M,\mathbf{r}' \models_x C_i(a)$ for an extension $\mathbf{r}'$ of $\mathbf{r}$ up to t'?
  - $M,\mathbf{r} \models_x^{\infty} C_i(a)$: does $M,\mathbf{r} \models_x^{t'} C_i(a)$ for some t'?

Reasoning: Complexity

- **Monitoring** and **Prediction** with bounded horizon $(M, \mathbf{r} \models_x^{t'} C_i(a))$ is decidable, but intractable in general
- **Unbounded Prediction** $(M, \mathbf{r} \models_x^{\infty} C_i(a))$ is undecidable in general, due to

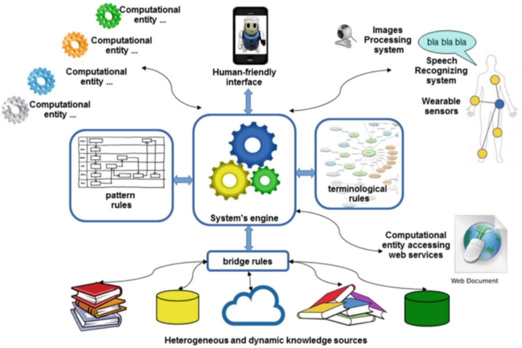
  (U1) kb_i, (U2) unbounded streams, (U3) pathologic window evaluation
- **Prediction** is in PSpace, if
  - each kb_i remains in *polynomial size*,
  - context evaluation (br_i, mng_i, $\mathbf{ACC}_i$) feasible in *polynomial space*,
  - the bridge rules br_i are *plain* (roughly, time references at eval time are fixed offsets) and all windows are *regular* (e.g., small time/tuple-based windows)

  $\Rightarrow$ streams manageable in pol. space (intuitively, recent input)
- Tractability (moreover logspace feasibility) needs severe restrictions
- Links to work on streams in databases [Gurevich *et al.*, 2007] and ontologies [Özçep, 2017], and recent work at Oxford and Linköping

MCS for Smart Cyber-Physical Systems (CPS)

- Distributed systems with lots of sensors
- Possible embedding in the Internet of Everything
- "Friendly & Kind" (F&K) systems, e.g. in e-health
- Bridge rules as vital elements for knowledge exchange
- mMCS are nicely abstract, but limitations require extensions; cf. [Costantini and Gasperis, 2016], [Cabalar *et al.*, 2017]

[Costantini and Gasperis, 2016]



- Proposals:
  - *dynamic mMCS (dMCS)* [Costantini and Gasperis, 2016], [Dao-Tran *et al.*, 2011]: link at runtime
  - *timed mMCS (tMCS)* [Cabalar *et al.*, 2017]: update state prior to bridge rule evaluation

MCS for Smart Cyberphysical Systems, cont'd

MCS Rationale

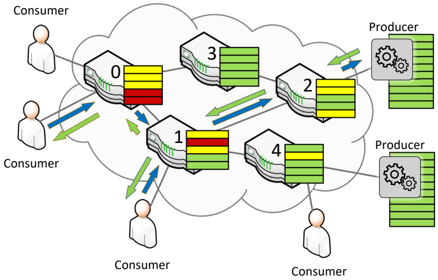
- stay at the abstract level
- use MCS more as *modeling tool*
    - heterogeneous components as contexts
    - interlinkage and exchange
- also useful for simulation

Scenario: Dynamic Configuration

- go beyond monitoring / prediction
- dynamically change / adapt the behavior of components

Content-Centric Networks

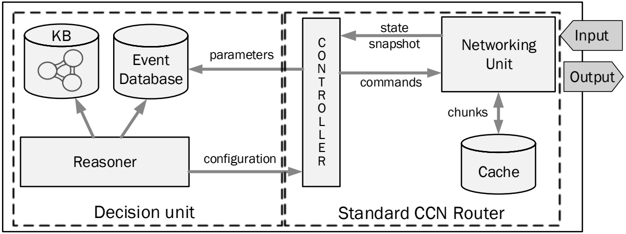
- The content in the network is addressed by “name” – physical location is irrelevant
- **Content-Centric Routers (CCR)** can route interest packages, cache and adapt media chunks in highly dynamic conditions
- Cache sizes are limited – efficient caching strategies needed



Example

- Factor: *Current daytime*
  - Morning: few users interested in different media
  - Evening: many users are watching a small amount of popular series
- Possible caching strategies for the scenarios above:
  - Random: replaces a *random chunk* in the cache with a random recent chunk
  - LFU: a new chunk replaces the *Least Frequently Used* chunk

Extended CCR – Intelligent Caching Agent



- **Legacy components of a CCR:**
  - networking unit: implements network interfaces
  - controller: manages content adaptation, routing and caching
- **Extended architecture:**
  - KB system: choose controller's decision making strategy
  - desired: human-readable KR language for admin actions

CCR Administration Problem

- LARS Encoding

$$
\begin{aligned}
&\text{high} \leftarrow \text{value}(V), \boxplus^{k\ \text{sec}} @_T\, \text{alpha}(V), 18 \leq V.\\
&\text{mid} \leftarrow \text{value}(V), \boxplus^{k\ \text{sec}} @_T\, \text{alpha}(V), 12 \leq V < 18.\\
&\text{low} \leftarrow \text{value}(V), \boxplus^{k\ \text{sec}} @_T\, \text{alpha}(V), V \leq 12.\\
&\text{lfu} \leftarrow \boxplus^{k\ \text{sec}} \Box\, \text{high}.\\
&\text{lru} \leftarrow \boxplus^{k\ \text{sec}} \Box\, \text{mid}.\\
&\text{fifo} \leftarrow \boxplus^{k\ \text{sec}} \Box\, \text{low}, \boxplus^{[k\ \text{sec}]} \Diamond\, \text{rtm50}.\\
&\text{done} \leftarrow \text{lfu} \vee \text{lru} \vee \text{fifo}.\\
&\text{random} \leftarrow \text{not done}.
\end{aligned}
$$

- A simple prototype, using `ndnSIM` (a general network simulator) and `solver.hex` (implements a LARS fragment usng dlvhex (hybrid ASP) was done

- Later, LARS Ticker [Beck *et al.*, 2017] encodings (for $k = 3$):

```
high :- value(V), alpha(V) at T [3 sec], 18 <= V.
mid :- value(V), alpha(V) at T [3 sec], 12 <= V, V < 18.
low :- value(V), alpha(V) at T [3 sec], V <= 12.
lfu :- high always [3 sec].
lru :- mid always [3 sec].
fifo :- low always [3 sec], rtm50 [3 sec].
done :- lfu.
done :- lru.
done :- fifo.
random :- not done.
value(5). value(15). value(25).
```

DynaCon: Dynamic Knowledge-Based (Re)configuration of Cyber-Physical Systems

Use Cases


traffic control
(Siemens)


power distribution grid
(Kelag)


network threat mgmnt
(Net4You)


rail transportation mgmnt
(LTE)

DynaCon: Dynamic Knowledge-Based (Re)configuration of Cyber-Physical Systems

Use Cases

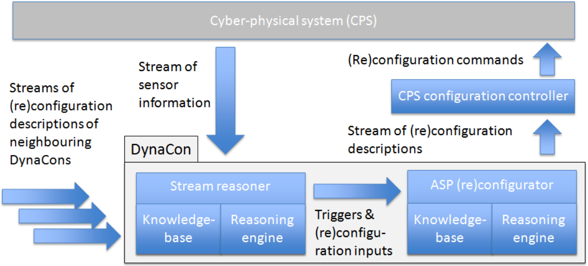

traffic control (Siemens)


power distribution grid (Kelag)


network threat mgmnt (Net4You)


rail transportation mgmnt (LTE)

Idea



DynaCon: Dynamic Knowledge-Based (Re)configuration of Cyber-Physical Systems

Use Cases


traffic control (Siemens)

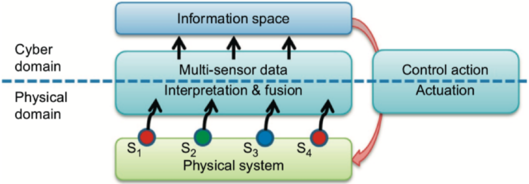

power distribution grid (Kelag)


network threat mgmnt (Net4You)


rail transportation mgmnt (LTE)

Idea

[Sarkar, 2011]



CPS: combing physical space with information space via computing, communication and control.

Dynamic Configuration as MCS

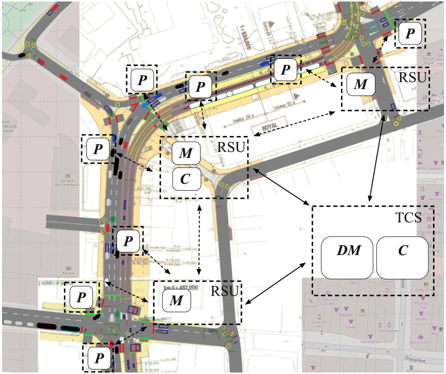
Observation: the MCS framework is suited to model dynamic configuration scenarios

- structuring into interlinked components, evolving over time
- modeling interlinkage through bridge rules
- logical *separation of concerns (SoC) / tasks*
  - **Producers**: contexts that produce information / output
    E.g. sensors can be viewed as such
  - **Monitors**: contexts that observe and aggregate data streams from producers, and report (feed information) to configurators
  - **Configurators**: contexts calculating the setup thru re-configuring the CPS; may involve complex decision component, richer high level stream reasoning
  - **Actuators**: contexts that change the setup in the CPS environment according to the output of the configurators

SoC may be weakened (integrate actuators into producers)

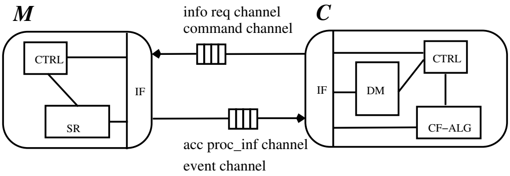
Scenario: Cooperative Intelligent Transporation Systems

- **Infrastructure as a CPS:**
  - communication via V2X
  - roadside units (RSU) at intersections
  - traffic participants are mobile sensors
  - central traffic control center (TCS) is connected to all RSUs
- **Producers:**
  - vehicles send their status
  - traffic lights send signal phases
- **Monitors:**
  - stream aggr./event detection on RSUs
  - high speed + volume sensor streams
  - local view of traffic
- **Configurators:**
  - configurator is in the TCS
  - optimize traffic flow via dynamic configurating of the traffic lights
  - global view of traffic
- **Actuators:**
  - on board of RSU



Intersection in Luxemburg

Component Interface



- **Monitor's concern:**

  Making the (variable data rate) input from the CPS accessible to the configurator by (i) detecting event, (ii) discretizing and accumulating data streams, (iii) sending the results via channels with limited data rate.

- **Configurator's Information Channels**
  - sending information:
    - *Command Channel*
    - *Information Request Channel*
  - receiving information
    - *Event Channel*
    - *Accumulated Process Information Channel*

- Separate: **Configuration Channel**
  - includes *adaptive monitoring*

Monitor vs. Configurator: Interface, cont'd

Event representation

- messages $m_e = (e, a, t, l, d, p)$,
- datalog encoding
  `event(eventType,sourceID,targetID,locationID,time,parameter).`

Process information messages

- messages with tuples $m_p = (i, a, t, l, p, u)$, $p = (d, v)$
- datalog encoding
- `information(infoType,sourceID,targetID,locationID,time, value,unit).`

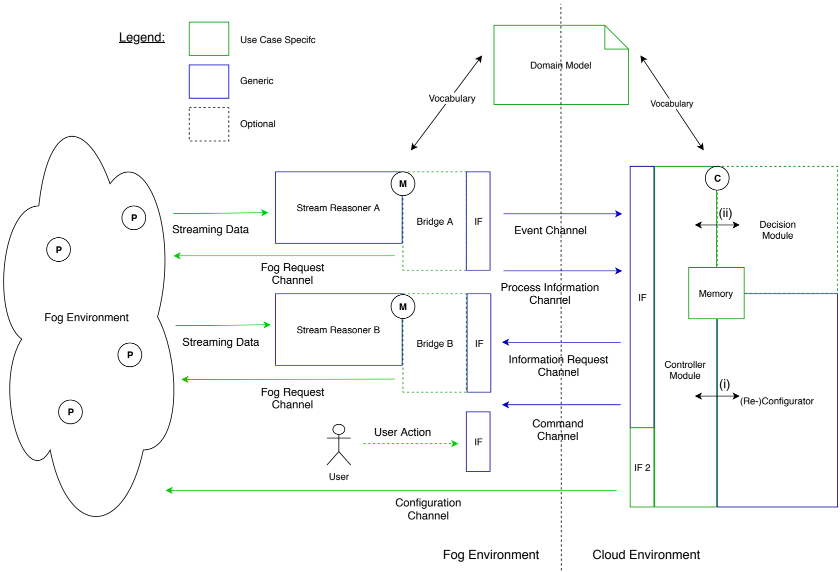
Commands

- Set parameter (Parameter, Value)
- Get parameter (Parameter)
- Reset
- Activate/deactivate rules or queries
- Update knowledge base (Update Operation)

datalog encodings

- `command(reset,sourceID,targetID).`
- `command(setParameter,sourceID,targetID,parameterID, <filter>,value).`

Refined DynaCon Architecture



Distributed Stream Processing

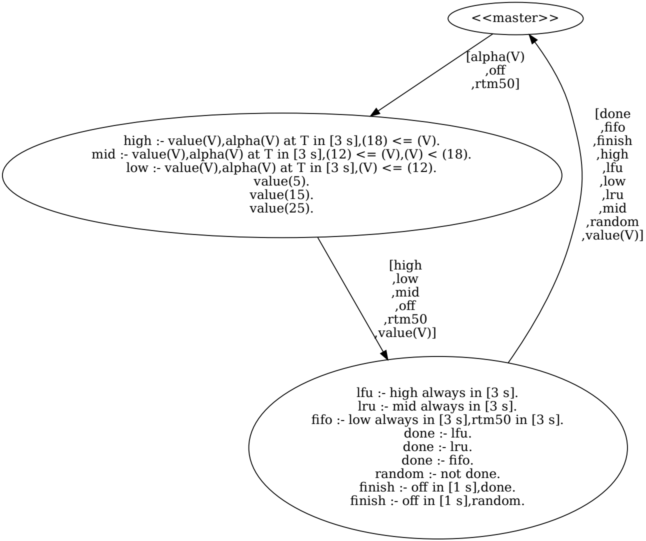
- LARS engines Ticker [Beck *et al.*, 2017], Laser [Bazoobandi *et al.*, 2017]: monolithic evaluation using a clock (ticks)
- performance issues under load
- as in stream processing, distribute computation

Distributed LARS (Outline):

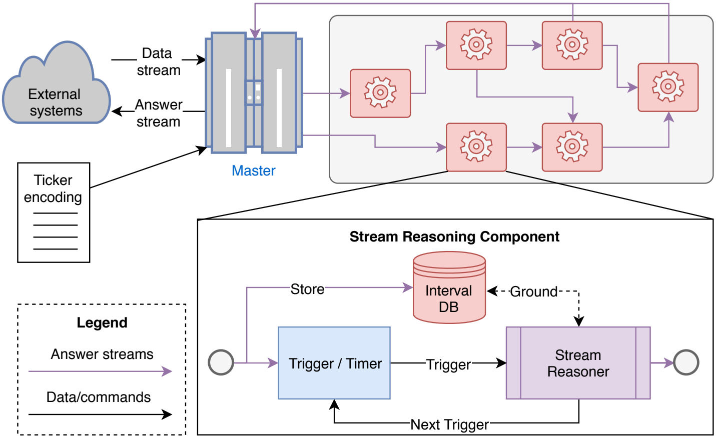
- streaming atoms: $a \mid @_{t'} a \mid \boxplus @_{t'} a \mid \boxplus \Diamond a \mid \boxplus \Box a$

  cast *time-point* to *interval* semantics (support *triggers*)
- decompose program P using an (stream) *dependency graph*
- a *component graph* over it yields a network of subprograms $P_1, \ldots, P_m$
  - each P_i is run by a stream reasoner
    - publishes streaming atoms to its successors,
    - requests streaming atoms from its predecessors (for itself or successors)
  - a special *master node* interfaces the outside world (publishes all externaal atoms, wants all internal atoms)
- *stream-stratification* (no cycle through windows) ensures a data pipeline

Component Graph



Distributed Stream Reasoning System



Master: computes the component graph and spawns nodes in the network

Conclusion

Summary

- MCS as versatile formalism, many extensions
- streaming data as an increasing computation setting
  - cf. [Ellmauthaler, 2018] for MCS and streaming
- Cyber-Physical Systems (CPS) as application area of MCS
- DynaCon: dynamic configuration, a challenging need in CPS
- distributed stream reasoning: LARS
  - BigSR, Strider [Ren, 2018]: hybrid adaptive distributed RSP engine, compile into Apache Spark / Flink

Issues and Ongoing/Future Work

- picture the role of MCS
- refined complexity
  - communication, memory, parallelization
- component interface languages
- adaptive monitoring: control language
- develop distributed LARS

References I

Chitta Baral and V. S. Subrahmanian.
Stable and extension class theory for logic programs and default logics.
J. Autom. Reasoning, 8(3):345–366, 1992.

Hamid R. Bazoobandi, Harald Beck, and Jacopo Urbani.
Expressive stream reasoning with laser.
In Claudia d'Amato, Miriam Fernández, Valentina A. M. Tamma, Freddy Lécué, Philippe Cudré-Mauroux, Juan F. Sequeda, Christoph Lange, and Jeff Heflin, editors, *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, volume 10587 of *Lecture Notes in Computer Science*, pages 87–103. Springer, 2017.

Harald Beck, Minh Dao-Tran, Thomas Eiter, and Michael Fink.
Towards a logic-based framework for analyzing stream reasoning.
In Irene Celino, Oscar Corcho, Daniele Dell'Aglio, Emanuele Della Valle, Markus Krötzsch, and Stefan Schlobach, editors, *Proceedings 3rd International Workshop on Ordering and Reasoning (Ordring 2014), October 19-20, 2014 Riva del Garda, Trentino, Italy*, number 1303 in CEUR Workshop Proceedings, pages 11–22. CEUR-WS.org, 2014.
Online `http://ceur-ws.org/Vol-1303/`.

Harald Beck, Minh Dao-Tran, and Thomas Eiter.
Answer update for rule-based stream reasoning.
In Q. Yang and M. Wooldridge, editors, *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15), July 25-31, 2015, Buenos Aires, Argentina*, pages 2741–2747. AAAI Press/IJCAI, 2015.

Harald Beck, Minh Dao-Tran, Thomas Eiter, and Michael Fink.
LARS: A logic-based framework for analyzing reasoning over streams.
In Blai Bonet and Sven Koenig, editors, *Proceedings 29th Conference on Artificial Intelligence (AAAI '15), January 25-30, 2015, Austin, Texas, USA*, pages 1431–1438. AAAI Press, 2015.

References II

- Harald Beck, Thomas Eiter, and Christian Folie.
  Ticker: A system for incremental asp-based stream reasoning.
  TPLP, 17(5-6):744–763, 2017.
- Harald Beck, Minh Dao-Tran, and Thomas Eiter.
  LARS: A logic-based framework for analytic reasoning over streams.
  Artif. Intell., 261:16–70, 2018.
- Gerhard Brewka and Thomas Eiter.
  Equilibria in heterogeneous nonmonotonic multi-context systems.
  In *AAAI*, pages 385–390, 2007.
- G. Brewka, F. Roelofsen, and L. Serafini.
  Contextual default reasoning.
  In *International Joint Conference on Artificial Intelligence (IJCAI 07)*, 2007.
- Gerhard Brewka, Thomas Eiter, Michael Fink, and Antonius Weinzierl.
  Managed Multi-Context Systems.
  In *IJCAI*, pages 786–791, 2011.
- Gerhard Brewka, Stefan Ellmauthaler, and Jörg Pührer.
  Multi-Context Systems for Reactive Reasoning in Dynamic Environments.
  In *ECAI*, pages 159–164, 2014.
- Gerhard Brewka, Stefan Ellmauthaler, Ricardo Gonçalves, Matthias Knorr, João Leite, and Jörg Pührer.
  Reactive multi-context systems: Heterogeneous reasoning in dynamic environments.
  Artif. Intell., 256:68–104, 2018.

References III

- Pedro Cabalar, Stefania Costantini, and Andrea Formisano.
  Multi-context systems: Dynamics and evolution.
  In Bart Bogaerts and Amelia Harrison, editors, *Proceedings of the 10th Workshop on Answer Set Programming and Other Computing Paradigms co-located with the 14th International Conference on Logic Programming and Nonmonotonic Reasoning, ASPOCP@LPNMR 2017, Espoo, Finland, July 3, 2017.*, volume 1868 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.
- Stefania Costantini and Giovanni De Gasperis.
  Bridge rules for reasoning in component-based heterogeneous environments.
  In José Júlio Alferes, Leopoldo E. Bertossi, Guido Governatori, Paul Fodor, and Dumitru Roman, editors, *Rule Technologies. Research, Tools, and Applications - 10th International Symposium, RuleML 2016, Stony Brook, NY, USA, July 6-9, 2016. Proceedings*, volume 9718 of *Lecture Notes in Computer Science*, pages 97–112. Springer, 2016.
- Minh Dao-Tran and Thomas Eiter.
  Streaming multi-context systems.
  In Carles Sierra, editor, *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17), August 19-25, 2017, Melbourne, Australia*, pages 1000–1007. AAAI Press/IJCAI, 2017.
- Minh Dao-Tran, Thomas Eiter, Michael Fink, and Thomas Krennwallner.
  Dynamic distributed nonmonotonic multi-context systems.
  In Gerd Brewka, Miroslaw Truszczyński, and Victor Marek, editors, *NonMon@30: Thirty Years of Nonmonotonic Reasoning*, pages 63–88. College Publications, London, UK, 2011.
- Thomas Eiter, Mustafa Mehuljic, Christoph Redl, and Peter Schüller.
  User guide: dlvhex 2.x.
  Technical Report INFSYS RR-1843-15-05, Institut für Informationssysteme, Technische Universität Wien, A-1040 Vienna, Austria, September 2015.

References IV

Thomas Eiter, Tobias Kaminski, Christoph Redl, Peter Schüller, and Antonius Weinzierl.
Answer set programming with external source access.
In Giovambattista Ianni, Domenico Lembo, Leopoldo E. Bertossi, Wolfgang Faber, Birte Glimm, Georg Gottlob, and Steffen Staab, editors, *Reasoning Web. Semantic Interoperability on the Web, 13th International Summer School 2017, London, UK, July 7-11, 2017, Tutorial Lectures*, number 10370 in LNCS, pages 204–275. Springer, 2017.

Stefan Ellmauthaler and Jörg Pührer.
Asynchronous multi-context systems.
In Thomas Eiter, Hannes Strass, Miroslaw Truszczynski, and Stefan Woltran, editors, *Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation - Essays Dedicated to Gerhard Brewka on the Occasion of His 60th Birthday*, volume 9060 of *Lecture Notes in Computer Science*, pages 141–156. Springer, 2015.

Stefan Ellmauthaler.
Multi-Context Reasoning in Continuous Data-Flow Environments.
PhD thesis, Leipzig University, Germany, 2018.
`http://nbn-resolving.de/urn:nbn:de:bsz:15-qucosa2-214577`.

Chiara Ghidini and Fausto Giunchiglia.
Local models semantics, or contextual reasoning = locality + compatibility.
Artificial Intelligence, 127(2):221–259, 2001.

F. Giunchiglia and L. Serafini.
Multilanguage hierarchical logics, or: How we can do without modal logics.
Artificial Intelligence, 65(1):29–70, 1994.

Ricardo Gonçalves, Matthias Knorr, and João Leite.
Evolving Multi-Context Systems.
In *ECAI*, pages 375–380, 2014.

References V

Yuri Gurevich, Dirk Leinders, and Jan Van den Bussche.
A theory of stream queries.
In Marcelo Arenas and Michael I. Schwartzbach, editors, *Database Programming Languages, 11th International Symposium, DBPL 2007, Vienna, Austria, September 23-24, 2007, Revised Selected Papers*, volume 4797 of *Lecture Notes in Computer Science*, pages 153–168. Springer, 2007.

Harald Beck, Minh Dao-Tran, and Thomas Eiter.
Contrasting RDF stream processing semantics.
In Guilin Qi, Kouji Kozaki, Jeff Z. Pan, and Siwei Yu, editors, *Semantic Technology – 5th Joint International Semantic Technology Conference (JIST 2015), November 11-13, 2015, Yichang, China, Revised Selected Papers*, number 9544 in LNCS, pages 289–298. Springer International Publishing Switzerland, 2015.

Özgür L. Özçep.
Representation Theorems in Computer Science A Treatment in Logic Engineering.
PhD thesis, Universität zu Lübeck, Germany, 2017.
Habilitation Thesis.

Xiangnan Ren.
Distributed RDF Stream Processing and Reasoning.
PhD thesis, Université Paris-Est, France, 2018.
`https://tel.archives-ouvertes.fr/tel-02083973/document`.

F. Roelofsen and L. Serafini.
Minimal and absent information in contexts.
In *Proc. IJCAI-05*, 2005.

S. Sarkar.
Autonomous Perception and Decision Making in Cyper-Physical Systems.
PhD thesis, Penn State University Graduate School, PA 16802, USA, 2011.

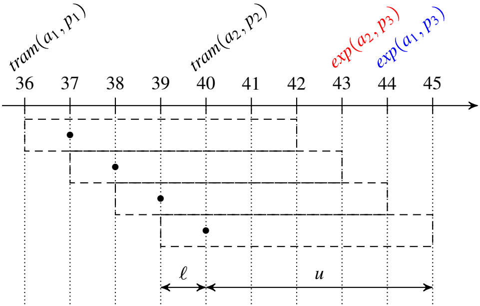
Data Snapshots: Window Functions

- Important aspect of stream processing: use only *window* view of data, i.e., limited observability at each point in time
- Different types of windows in practice:
  - *time-based windows* (within time bounds)
  - *tuple-based windows* (number of tuples, count)
  - *partition-based windows* (split input data, process separately)
  - in addition, *sliding* or *tumbling* (consider atom repeatedly / once)
- Model data snapshots (windows) as *substreams* of a stream
- Formally, windows are functions

$$w : (S, t) \mapsto S'$$

assigning each stream $S = (T, \upsilon)$ and $t \in T$ a substream $S' \subseteq S$, which means $S' = (T', \upsilon')$ such that $T' \subseteq T$ and $\upsilon'(t) \subseteq \upsilon(t)$, for all $t \in T'$

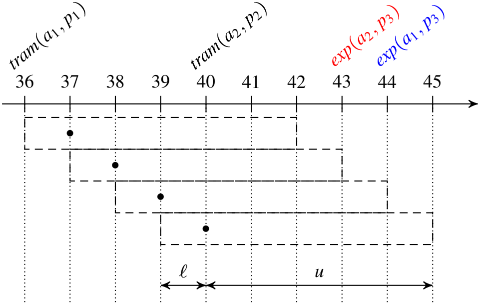
Window Functions: Example



$$S' = w(S, t)$$

Window Functions: Example

time-based window $w_\tau^{1,5(1)}$, looking back 1 and forward 5 steps (at most), with step size 1 (i.e., *sliding*)



$$S' = w_\tau^{1,5,1}(S, 40) = ([39, 45], \left\{ \begin{array}{l} 40 : \{tram(a_2, p_2)\}, \\ 43 : \{exp(a_2, p_3)\}, \\ 44 : \{exp(a_1, p_3)\} \end{array} \right\})$$

LARS Formulas

LARS language: extend logic language stream access / processing

- Atoms from $\mathcal{A}$ (atomic formulas a)
- Boolean connectives $\wedge$, $\vee$, $\rightarrow$, $\neg$
- Window operators $\boxplus$ (substream generation), $\triangleright$ (reset to original stream)

$$\boxplus^{w} \iff w(S,t)$$

Examples

- $\boxplus_{\tau}10 := \boxplus^{w_{\tau}^{10,0(1)}}$ last 10 units (sliding time-based)
- $\boxplus_{\tau}{+}5 := \boxplus^{w_{\tau}^{0,5(1)}}$ next 5 units
- $\boxplus_{\#}n = \boxplus^{w_{\#}^{n,0}}$ last n tuples (sliding tuple-based window)

- Temporal operators $\Diamond$, $\Box$, $@_t$

$$@_{20}\,tram(a_2,p_1) \qquad \boxplus_{\tau}{+}5\,\Diamond exp(a_1,p_3)$$

- Note: nesting of windows is possible!

$$\boxplus_{\tau}60\Box\,\boxplus_{\tau}\,5\Diamond tramAt(p_1) \qquad \boxplus_{\#}\,n\,\boxplus_{\tau}\,5\Diamond tramAt(p_1)$$

Regular Windows & Plain Bridge Rules

Regular Windows

- pathologic window functions $w(S,t)$ may e.g. interpret t as Gödel number of a computation; thus we hit undecidability.
- A window function $w(S,t)$ is *regular*, if
  - (i) $w(S,t)(t')$, i.e., the data in the window $w(S,t)$ at time t', depends only on S from $t', t'+1$ etc. onwards (allows for *data dropping*)
  - (ii) for some $l \geq p \geq 0$ polynomial in $|kb_i(0)|$, we have $w(S,t) = w(S', t+p)$ for every t and streams S, S' that coincide on the past (future) l time points around t resp. $t+p$ *having data* (informally, w is *periodic* and l is a *limit for evaluation*)
- small (polynomial-size) time-based, tuple-based windows are regular.

Plain Bridge Rules

- simply memorizing the data within the limit *with their actual time points* is not feasible under a space constraint wrt. $|kb_i(0)|$
- a (schematic) bridge rule is *plain*, if time references are to evaluation time ($\boxplus^0 @_Z \top$) with fixed offset os ($Z \pm os$)
- For plain bridge rules (cf. running example), full memorization can be avoided

Lemma. If br_i is plain and any window occurring in it is regular, a sufficient fragment of each input stream S_{ki} to evaluate br_i can be maintained in polynomial space.

MCS limitations for Smart CPS

- **Issues** [Costantini and Gasperis, 2016], [Cabalar *et al.*, 2017]

  - *Grounded (Propositional) Knowledge*

    ⇒ expand initial grounding of open rules gradually (fixpoint)
  - *Logical Omniscience and Unbounded Resources*

    ⇒ delay bridge rule application *with commitment*
  - *Update Problem (by Environment)*

    ⇒ environment update prior to mngmt update (tMCS)
  - *Full System Knowledge*

    ⇒ look up yellow pages for neighbors
  - *Static System*

    ⇒ yellow pages of current contexts (cf. dynamic configuration [Dao-Tran *et al.*, 2011])
  - *Unique Source*

    ⇒ dynamic name binding: pick suitable context (by role)
  - *Uniform Knowledge Representation Format*

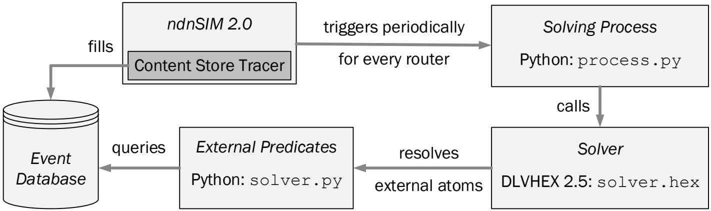
    ⇒ model / KB alignment
  - *Equilibria Computation and Consistency Check*

    black box (privacy) vs glass box contexts (efficiency)

- **Proposal:** *dynamic mMCS (dmCS)*, special contexts (e.g. yellow pages)

- **But:** formalization amenable to analysis ?? (cf. aMCS)

CNN: Framework Implementation



- `ndnSIM`: a general network simulator
- `solver.hex`: implements a LARS fragment using the dlvhex solver (hybrid ASP)
- `solver.py`: comprises implementation of external predicates, e.g.
  - $alpha/1$: returns the estimated $\widehat{\alpha}$ value of the Zipf distribution
- Later: Ticker implementation