Combining Event Calculus and Description Logic Reasoning via Logic Programming

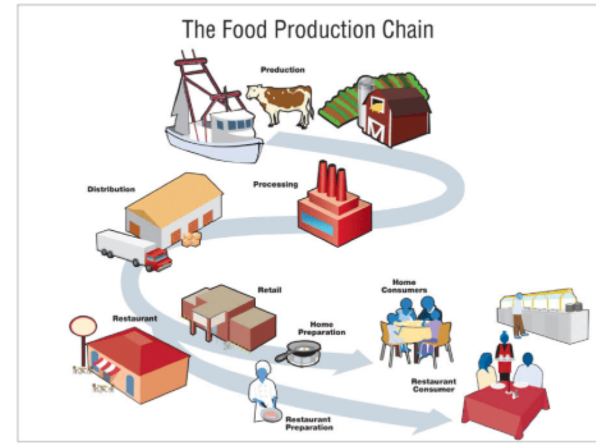
Peter Baumgartner

Data61|CSIRO and ANU, Canberra, Australia

Project Background: A Logic Based System for Situational Awareness

Situational awareness ≈ comprehending system state as it evolves over time

Example: Food Supply Chain



- Are goods delivered within 3 hours and stored below 25°C?
- Why is the truck late?
- What is the expected quality (shelf life) of the goods?

What's the problem?

- **Multiple aspects**: temporal/causal/structural/physical/… ← Logic program
- Events **happened** ≠ events **reported** (errors, incomplete, late …) ← Belief revision
- **Uncertainty**: **multiple** plausible explanations for given facts ← Models

This Work

- More expressive modelling language for better domain modelling
- Extension 1: Description logic interface
- Extension 2: Event calculus
- Implementation in Fusemate system

Fusemate - Language and Model Computation Overview

[IJCAR 2020]

Input language: Prolog-like rules

R(a,b)

R(X,Y) :- R(Y,X)

R(X,Z) :- R(X,Y), r(Y,Z)

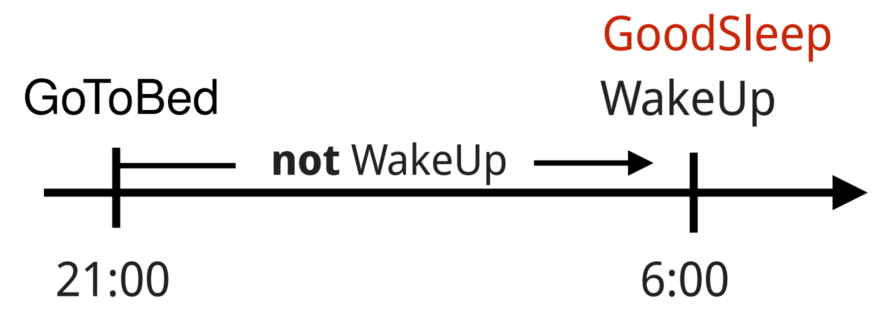
Models
R(a,b)
R(b,a)

Botton-up procedures
(Hyper tableau, Hyper resolution, …)

Default negation: stratification "by time"

GoodSleep(time) :-
  WakeUp(time),
  GoToBed(t), t <= time - 8,
  not (t <= s, s < time, WakeUp(s))

"not" subgoals must be strictly earlier "<" than current time



~~unhappy(time) :- Now(time), **not** winLottery(time+7)~~

Disjunctions: possible model semantics [Sakama 90]

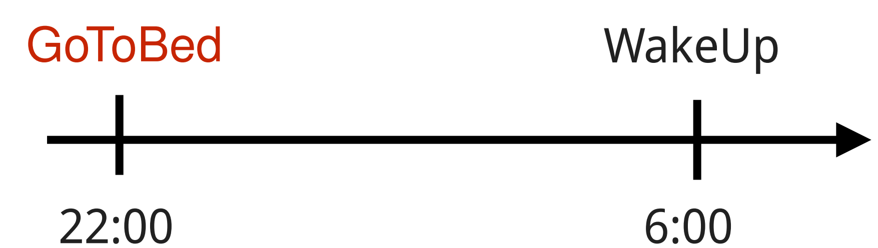
Thirsty(time) **or** Hungry(time) :- GoodSleep(time)

Models		Inclusive "or"
Thirsty(10)	Hungry(10)	Hungry(10)
		Thirsty(10)

Belief revision

fail(+ GoToBed(time - 8)) :-
  WakeUp(time),
  not (GoToBed(t), t <= time - 8)

Add retrospectively



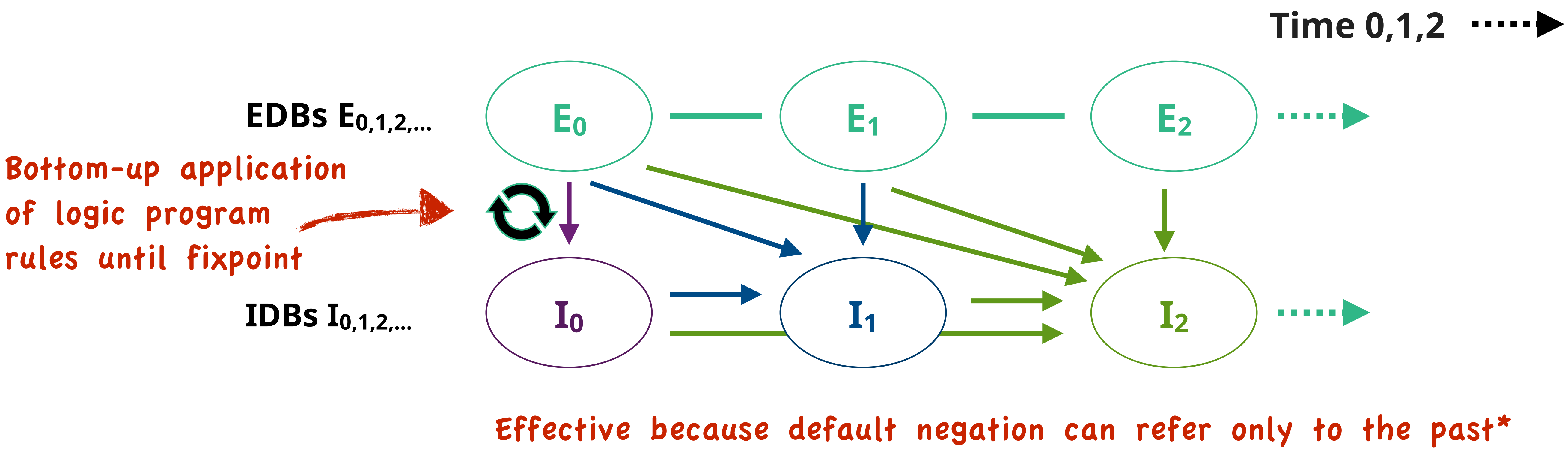
Application: Situational awareness = model computation

Stratified Model Computation

Modelling Setup for Situational Awareness

- EDB: Timestamped facts (“events”)　　　E0, E1, E2, …
- IDB: Models for derived predicates up to “now”

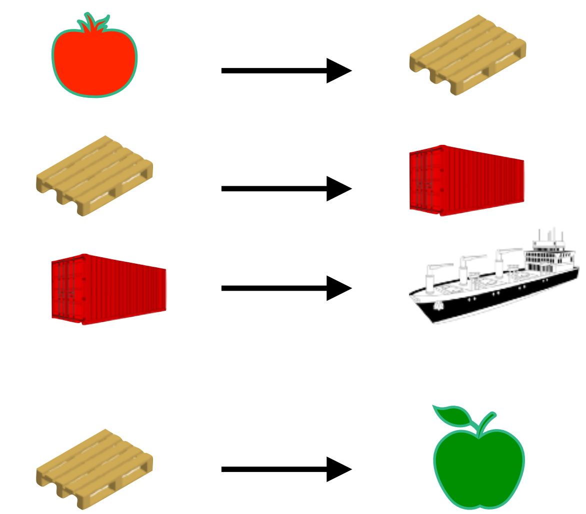
Model Computation



Revision

Revision = programmable addition/removal of events in the past + restart of model computation

Logic Program Example: Supply Chain



Derived "In" relation

```
In(time, obj, cont) :-
      Load(time, obj, cont)

 // In transitivity
 In(time, obj, cont) :-
      In(time, obj, c),
      In(time, c, cont)

 // Frame axiom for In
 In(time, obj, cont) :-
      In(prev, obj, cont),
      Step(time, prev),
      not Unload(prev, obj, cont),
      not (In(prev, obj, c), Unload(time, c, cont))
```

Integrity constraints and revision

```
// No Unload without earlier Load
fail :-
      Unload(time, obj, cont),
      not (Load(t, obj, cont), t < time))

// Unload a different object
fail(- Unload(time, obj, cont), + Unload(time, o, cont)) :-
      Unload(time, obj, cont),
      not (Load(t, obj, cont), t < time),
      Load(t, o, cont),
      t < time,
      SameBatch(t, b),
      ((b contains obj) && (b contains o))
```

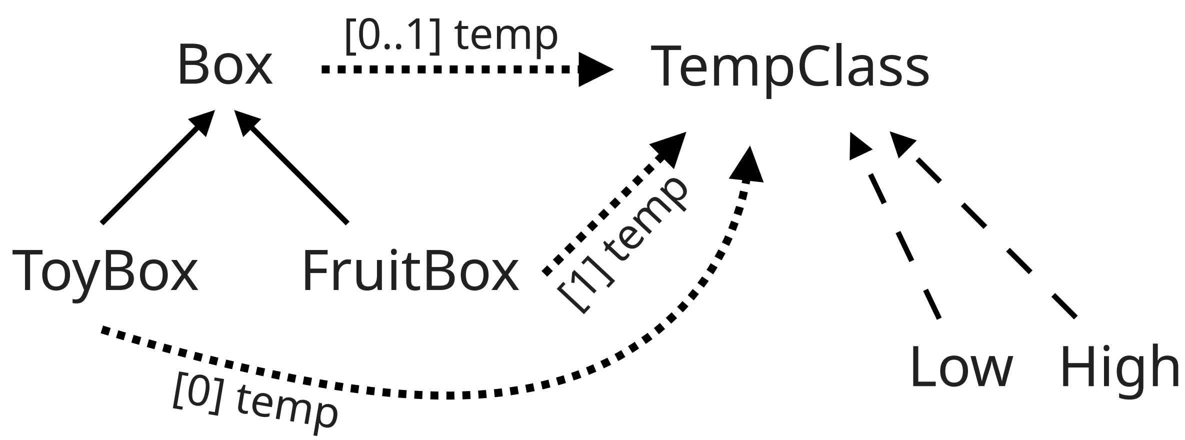
\+ *4 more rules*

Experience: Logic programs often
(a) are too low-level, and
(b) suffer from non-termination for "tuple generating dependencies"
-> Extend reasoning framework with Description Logic reasoning and Event Calculus

Description Logic Reasoner Interface

Description Logics

- A DL KB consists of a TBox (concept definitions) and an ABox (instance assertions)
- The concrete choice of DL is not important here, but must include ALC and satisfiablity must be decidable



KB = (ABox, TBox)

TBox

$\text{Box} \sqsubseteq \forall\, \text{temp.TempClass}$

$\text{FruitBox} \sqsubseteq \exists\, \text{temp.TempClass}$

$\text{ToyBox} \sqsubseteq \neg\exists\, \text{temp.TempClass}$

$\text{FruitBox} \sqsubseteq \text{Box}$

$\text{ToyBox} \sqsubseteq \text{Box}$

temp is a functional role

ABox

$\text{Low} : \text{TempClass}$

$\text{High} : \text{TempClass}$

$\text{Box}_0 : \text{FruitBox}$

$\text{Box}_1 : \text{FruitBox}$

$\text{Box}_2 : \text{Box}$

$\text{Box}_3 : \text{ToyBox}$

$\text{Box}_4 : \text{Box} \sqcap \forall\, \text{temp.}\neg\text{TempClass}$

$\text{Box}_5 : \text{Box} \sqcap \exists\, \text{temp.TempClass}$

Reasoning

- Does Box_0 have a temp attribute?
- Is Box_5 a FruitBox?
- Are FruitBox and ToyBox disjoint?
- Is (ABox, TBox) satisfiable?

[CADE-2021 SD]:
DL ALCIF by mapping to Fusemate disjunctive logic program + loop check

Description Logics + Logic Programming Approach - Overview

DL and LP are Complementary

Open world vs closed world, entailment vs models, unique name assumption no/yes

- Box_0 : FruitBox
- Box_1 : FruitBox
- Box_2 : Box
- Box_3 : ToyBox
- Box_4 : Box $\sqcap \forall$ temp.$\neg$TempClass
- Box_5 : Box $\sqcap \exists$ temp.TempClass

Here: Timed Setting

Time	10	20	30	40	50
Action	Load Box_0 Load Box_1	Load Box_2	Load Box_3 Load Box_4		Unload
Sensor	$Box_0 : -10°$	$Box_2 : 10°$	$Box_0 : 2°$	$Box_0 : 20°$	

t=50 Box_0 temp problem?

Truck cooling problem?

What boxes to check?

Goal: Understanding situation as it evolves over time

t=10
- Box0 has a known* low temp
- Box1 has some unknown temp
- Box2 is not known to have a temp
- Box3 is known to have no temp

t=20
- Box2 has a known high temp

t=30
- Box0 has a known high temp

* “known” = “follows wrt FOL”

Approach: DL+Rules(+Event Calculus)

- **DL**: black-box theory reasoner - can talk about implicitly exisiting individuals
- **EC**: actions and their effects over time - can add “from now on unless change” to above properties
- **Rules**: glue between DL+EC - can bring in concrete domains (numbers)

Description Logic Interface - Queries

DL Query Example (Body Literal)

(ABox(I, 20), TBox) |= Box_2 : FruitBox, (Box_2, Temp): High

ABox in interpretation I at time 20

DL Query Syntax

The following forms can be used in rule bodies

$T \models \vec{q}$	DLISSAT(T)	DLISUNSAT(T)
$(A,T) \models \vec{q}$	DLISSAT(A,T)	DLISUNSAT(A,T)

- T is a TBox
- A is an ABox, implicitly A(currentI, now) where
  $A(I,t) = \{a : C \mid a : C \ @ \ t \in I\} \cup \{(a,b) : R \mid (a,b) : R \ @ \ t \in I\}$
- $\overrightarrow{q}$ is a query, i.e., a sequence of terms representing an ABox
- $(A,T) \models \overrightarrow{q}$ means "$A \cup T \models \bigwedge \overrightarrow{q}$" as FOL formulas

TBox

Box $\sqsubseteq \forall$ temp.TempClass
FruitBox $\sqsubseteq \exists$ temp.TempClass
ToyBox $\sqsubseteq \neg\exists$ temp.TempClass
FruitBox $\sqsubseteq$ Box
ToyBox $\sqsubseteq$ Box

Interpretation I with timed ABox Assertions

Box_0 : FruitBox @ 10
:
Box_2 : Box @ 20
(Box_0, High): temp @ 20

Description Logic Interface - Examples

$Box \sqsubseteq \forall\, temp.TempClass$
$FruitBox \sqsubseteq \exists\, temp.TempClass$
$ToyBox \sqsubseteq \neg\exists\, temp.TempClass$
$FruitBox \sqsubseteq Box$
$ToyBox \sqsubseteq Box$

Box_0 : FruitBox @ 10
:
Box_2 : Box @ 20
(Box_0, High): temp @ 20

Materialization

Can derive new ABox assertions (even in the past)!

x : Box @ time :-
(x : _ @ time), // x is an individual in an ABox assertion at "time"
TBox |= x : Box // Implicit ABox (A(I, time)

Variables in DL Queries grounded now

Box has temp

TempBox(time, box) :-
box : Box @ time,
TBox |= box : ∃ Temp . TempClass

Box has known temp

KnownTempBox(time, box) :-
box : Box @ time,
temp ∈ { Low, High }, // Guess
TBox |= (box, temp): Temp

Box never had known high temp in the past

ColdBox(time, box) :-
box : Box @ time,
not (t < time,
(A(I, t), TBox) |= box : Box, (box, High) : Temp)

(Stratifed) DL call under default negation!

Description Logic Interface - Semantics

Query Evaluation

Reduce query evaluation to standard DL knowledge base satisfiability

$(A, T) \models a : C$ iff $(A \cup \{a : \neg C\}, T)$ is unsatisfiable

$(A, T) \models (a, b) : r$ iff $(A \cup \{a : \forall r . \neg B,\ b : B\}, T)$ is unsatisfiable, with B fresh

Stratification

- **Implicit ABox** $A(I, t)$ - use concept and role assertions timed t
- **Explicit ABox**: not automatically, use with care :)

See paper for details

Unique Name Assumption (UNA)

- **DL** does not assume UNA

  E.g. $A = \{(c, a) : r, (c, b) : r\}$ with functional r is satisfiable only if $I(a) = I(b)$
- **LP** does assume unique name assumption, i.e., $I(a) \neq I(b)$
- **Solution**: enforce UNA in DL by adding axioms

  E.g. $N = \{a, b, c\}$ are all current named ABox individuals

  Add to ABox $\{a : N_{ab},\ b : \neg N_{ab},\ a : N_{ac},\ c : \neg N_{ac},\ \ldots\}$ where N_{xy}'s are fresh concept names

Description Logic Interface - Soundness and Completeness

Model computation soundness and completeness rests on the following properties

Box2 : ∃ **temp . TempClass**

DL-safe rules

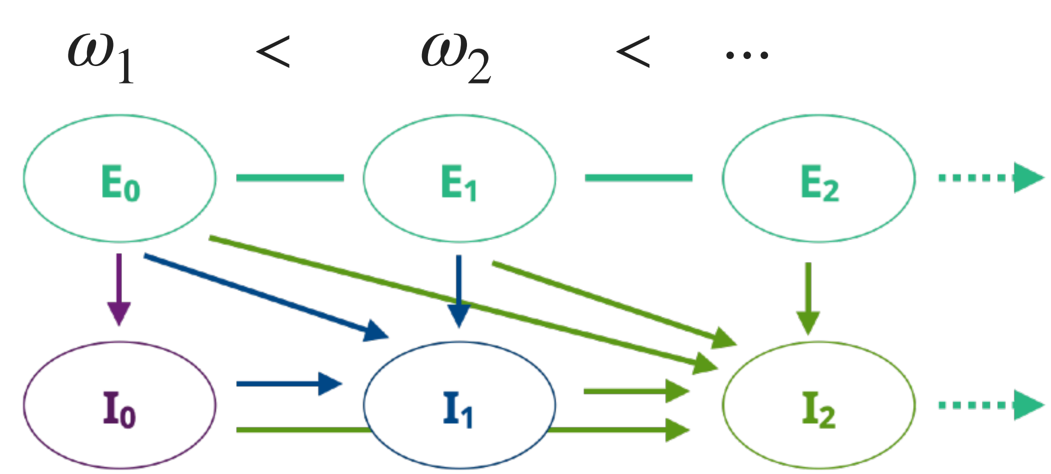
- **Named** individuals: those that appear explicitly in ABox assertions
- **Unnamed** individuals: implicitly constructed (Skolem)
- Rules are DL-safe: unnamed individuals cannot escape their query scope

Monotonicity

- Rules H :- B must be monotonic: if $I \models B$ and $J \supseteq I$ then $J \models B$
- No problem with stratified negation
- [OK] DL queries $T \models \overrightarrow{q}$ and DLISUNSAT(T) are always monotonic by monotonicity of FOL
- DL queries $(A, T) \models \overrightarrow{q}$, DLISUNSAT($A$,$T$), DLISSAT($T$) and DLISSAT($A$,$T$) use with care

Compactness

- Fixpoint model requires transfinite induction in general
- Not effective for aggregation operator $\{P(x,t) \mid Q(x,s),\ s < t\}$
- However not a problem because interest only in finite models
- (DL query evaluation always compact because of FOL)



Event Calculus

Event Calculus [Kowalski & Sergot 1986]

- The event calculus (EC) is a logical language for representing and reasoning about actions and their effects
- The formulation below follows the original logic program, with adaptions and extensions for DL

Actions and Fluents

- A **fluent** is a property that **HoldsAt** over a time period
- Fluents are **initiated** or **terminated** by **actions** that **happen** at given time point

Time	10	20	30	40	50
Action	Load Box_0	Load Box_2	Load Box_3		Unload
	Load Box_1		Load Box_4		
Sensor	$\text{Box}_0 : -10°$	$\text{Box}_2 : 10°$	$\text{Box}_0 : 2°$	$\text{Box}_0 : 20°$	

t=20	Load(Box_2)	initiates	OnTruck(Box_2)	HoldsAt(20, OnTruck(Box_2))	$\in I$
t=50	Unload	terminates	OnTruck(Box_i)	HoldsAt(50, OnTruck(Box_i))	$\notin I$

Problem Specific Axioms

Initiates(time, Load(box), OnTruck(box)) :-
    box : Box @ time

Terminates(time, Unload(box), OnTruck(box)) :-
    HoldsAt(time, OnTruck(box))

Problem Specific Events

Happens(20, Load(Box_2))

Happens(50, Unload)

EC Library

HoldsAt(time+1, f) :-
    Initiated(time, f),
    not Terminated(time, f)

HoldsAt(time, f) :-
    HoldsAt(time-1, f),
    not Terminated(time, f)

Event Calculus

Linking DL with EC

- Often, ABox assertions are meant to **hold over time** instead of **time points** only
- That is, timed ABox assertions can be fluents now

"From time 0 on"	vs	**"At time 0"**
HoldsAt(0, Box_5: Box ⊓ ∃ Temp . TempClass)	vs	Box_5: Box ⊓ ∃ Temp . TempClass @ 0

- Add axioms for turning ABox fluents into timed ABox assertions again (but not vice versa)

x : c @ time :-
HoldsAt(time, x : c)

(x, y): r @ time :-
HoldsAt(time,(x, y) : r)

Rule with ABox Fluent, Action and Concrete Data

Initiates(time, SensorEvent(box, temp), (box, High) : Temp) **and**
Terminates(time, SensorEvent(box, temp), (box, Low) : Temp)) :-
Happens(time, SensorEvent(box, temp)),
temp > 0

If box temp sensor > 0
then box temp is "high" from now on and no longer "low" from on

Conclusion

Contributions

- **Theoretical**: very liberal Rule + DL combination, conditions for soundness and completeness
- **KR language design**: extension of LP with DL + EC "very useful" for situational awareness

  Hard to quantify, but see paper for "complex" anomaly detection example
- **Implementation**: Fusemate https://bitbucket.csiro.au/users/bau050/repos/fusemate

Open Problem

- The **ramification problem** is concerned with indirect consequences of actions, such as **conflicts**
- It occurs in a pronounced way here
- Example: rule for terminating a box' temp fluent

  Terminated(time+1, (box, temp) : Temp)) :-

  RemoveTemp(time, box), // Some condition for removing box Temp

  (box, temp) : Temp @ time // Attribute to be removed
- This rule does not always work

  E.g, for a FruitBox the box' temp attribute is entailed by the "black box" TBox

$$\begin{aligned} \text{Box} &\sqsubseteq \forall\, \text{temp.TempClass} \\ \text{FruitBox} &\sqsubseteq \exists\, \text{temp.TempClass} \\ \text{ToyBox} &\sqsubseteq \neg\exists\, \text{temp.TempClass} \\ \text{FruitBox} &\sqsubseteq \text{Box} \\ \text{ToyBox} &\sqsubseteq \text{Box} \end{aligned}$$

temp is a functional role

- AFAIK "repairing" ABoxes is ongoing research but can be done in special cases