$\mathcal{M}$odular-$\mathcal{E}$: A Logic for Assimilating Unexpected Observations Along a time Line

Antonis Kakas (Cyprus) Loizos Michael (Harvard) Rob Miller (UCL)

part 1: Background and aims

part 2: Problems with existing approaches

part 3: $\mathcal{M}$odular-$\mathcal{E}$ syntax and semantics

part 4: Properties and extensions of $\mathcal{M}$odular-$\mathcal{E}$

part 5: Ideas for future work

Read more in LPNMR'05 and forthcomming AIJ papers

Historical Context

The idea of using symbolic logic to model human reasoning goes back a long way ...

- Before A.I. - Aristotle, Chrysippus, Descartes, Leibniz (a librarian!), de Morgan, Frege, Boole, Venn, Russell, Tarski, Gödel, ...

- A.I. - John McCarthy
  - "*Programs with Common Sense*" (1959).
  - Concepts such as "cause", "effect" and "action" are essential for key human reasoning processes such as *planning*.
  - In 1969 McCarthy and Hayes invent the *Situation Calculus*.
  - Broadens out into the A.I. sub-field of *Reasoning about Action*.

Modeling Cause and Effect in Logic

- Example Applications:
  - Automated planning - underpins much contemporary research.
  - Causal Modeling - e.g. simulation of complex biological systems.
  - Software Specification - e.g. of “event-driven” systems.

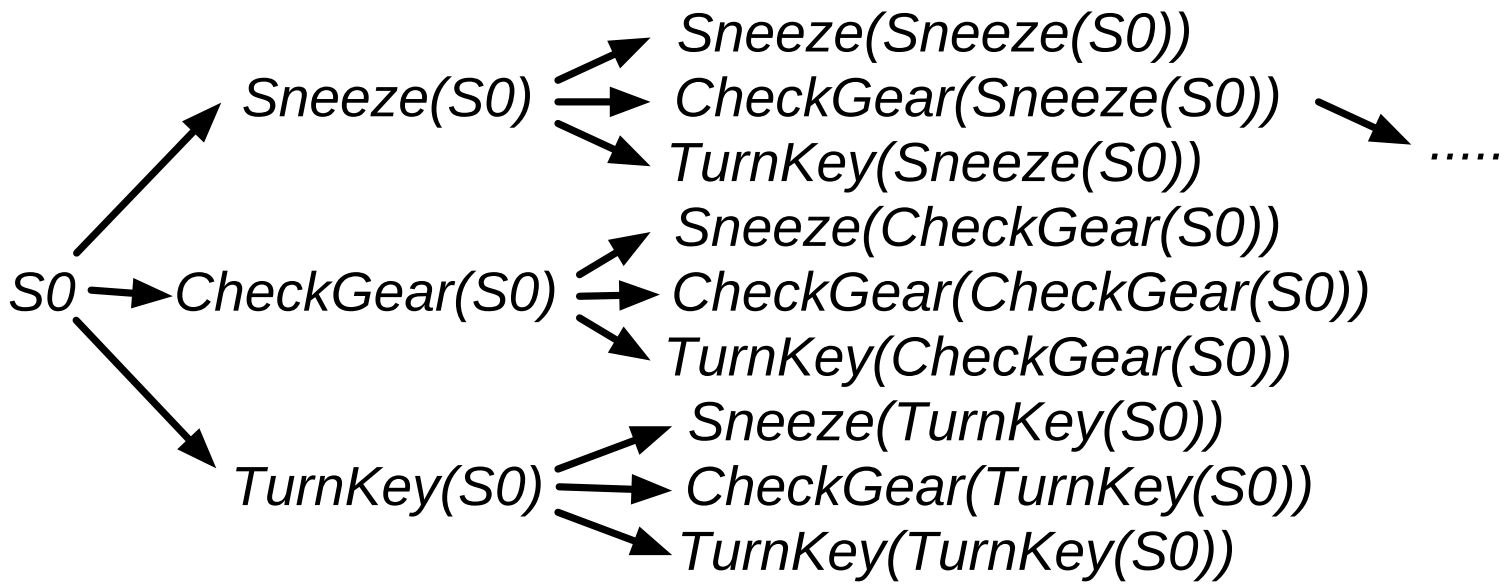
- Issues and Problems:
  - The Frame Problem - reasoning about non-effects of actions.
  - The Ramification Problem - reasoning about indirect effects.
  - The Qualification Problem - reasoning about unexpected effects.

Event Calculus, Language $\mathcal{E}$ and $\mathcal{M}$odular-$\mathcal{E}$

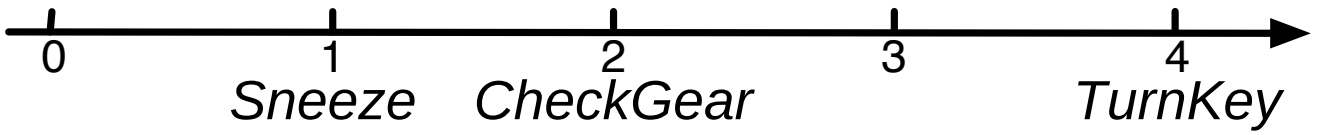
- Logic-based, model-theoretic formalisms for reasoning about action.
- Event Calculus first proposed by Bob Kowalski and Marek Sergot as a *logic program*, and later re-cast in *classical logic*.
- Language $\mathcal{E}$ and $\mathcal{M}$odular-$\mathcal{E}$ ($\mathcal{ME}$) are specialised "action description" logics developed from the Event Calculus, with a restricted high-level syntax.
- All similar in some respects to the Situation Calculus but with a different ontological basis, with:
  - *fluents* (properties that are true or false at different times),
  - *actions*,
  - *time-points* (arranged in a line), and
  - *events* (for a *narrative* of action "happenings" at particular times).

Representing Time

- Situation Calculus - branching, constructed from hypothetical actions:



- Event Calculus, $\mathcal{E}$ and $\mathcal{ME}$ - linear, events embedded within it:



An Example $\mathcal{M}$odular-$\mathcal{E}$ Theory

- $\{\mathit{TurnKey}, \mathit{Petrol}, \mathit{BatteryCharged}\}$ `causes` $\mathit{EngineRunning}$
  $\{\mathit{EngineRunning}, \neg\mathit{Oil}\}$ `causes` $\mathit{EngineTooHot}$
  $\mathit{EngineTooHot}$ `causes` Smoke

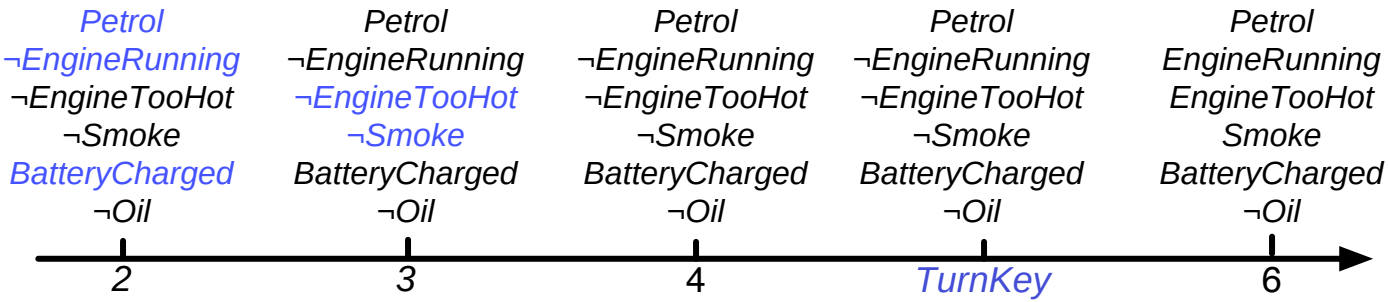
  $(\mathit{Petrol} \wedge \mathit{BatteryCharged} \wedge \neg\mathit{EngineRunning})$ `holds-at 2`
  $(\neg\mathit{EngineTooHot} \wedge \neg\mathit{Smoke})$ `holds-at 3`
  $\mathit{TurnKey}$ `occurs-at 5`

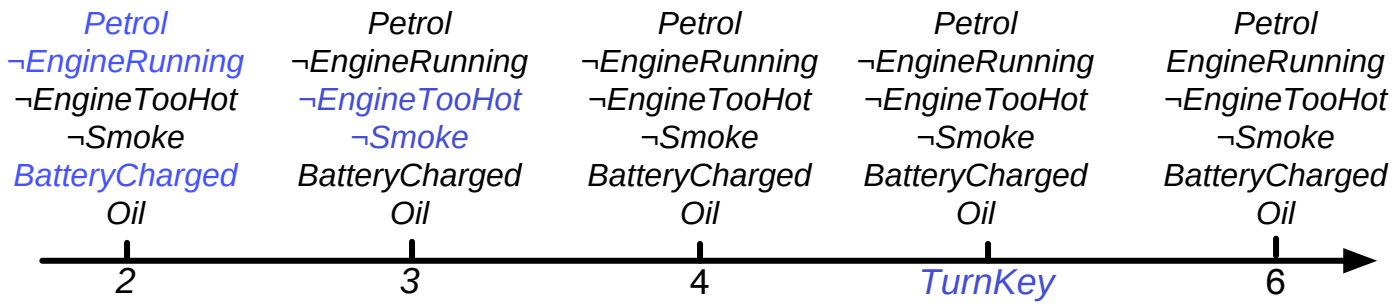
- Two models, because we don't know about *Oil*.

- Solution to the Ramification Problem ensures that if there's no oil, there's smoke e.g. at time 6.

Models of A $\mathcal{M}$odular-$\mathcal{E}$ Theory

$\{$*TurnKey*, *Petrol*, *BatteryCharged*$\}$ `causes` *EngineRunning*
$\{$*EngineRunning*, ¬*Oil*$\}$ `causes` *EngineTooHot*
EngineTooHot `causes` *Smoke*
(*Petrol* ∧ *BatteryCharged* ∧ ¬*EngineRunning*) `holds-at 2`
(¬*EngineTooHot* ∧ ¬*Smoke*) `holds-at 3`
TurnKey `occurs-at 5`





The Qualification Problem

"In order to fully represent the conditions for the successful performance of an action, an impractical and implausible number of qualifications would have to be included in the sentences expressing them."

– John McCarthy, 1980

$$\{\mathit{TurnKey}, \mathit{Petrol}, \mathit{BatteryCharged}, \neg \mathit{PotatoInExhaustPipe}, \ldots\ldots\}$$
$$\text{causes } \mathit{EngineRunning}$$

Aims of This Study

To to look at aspects of the *qualification problem* in the context of reasoning about *narratives* of action occurrences and observations:

- "Exogenous problem" – how to recover from unexpected observations:

  TurnKey `causes` *Running*
  TurnKey `occurs-at 1`
  ¬*Running* `holds-at 2 (!)`

- "Endogenous problem" – how to add qualifications, constraints and other effect laws in a modular and elaboration tolerant way:

  either *Broken* `prevents` *Running* (from being caused)
  or `always` ¬(*Broken* ∧ *Running*) (at this temporal granularity)
  or *Broken* `causes` ¬*Running* (a competing indirect effect law)

  (perhaps together with `normally` ¬*Broken*)

Original Strategy and Some Problems

We wanted to

- Introduce extra (default) reasoning mechanisms into the Language $\mathcal{E}$, in particular adapting some of Thielscher's AIJ ideas.

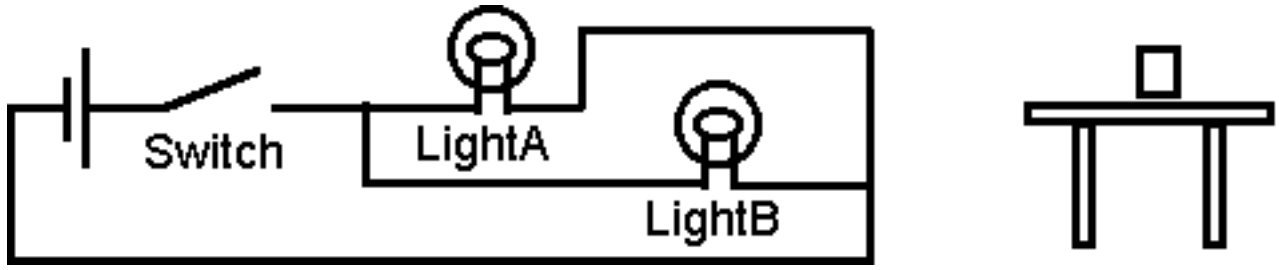
But ...

- Language $\mathcal{E}$'s solutions to the frame and ramification problems are not robust or complete enough for this.
- Other formalisms have similar or related deficiencies for our purposes.

So ...

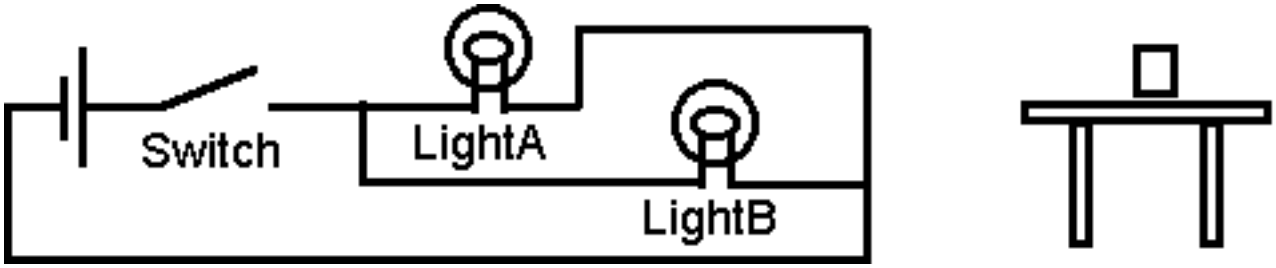
- We developed $\mathcal{M}$odular-$\mathcal{E}$ ($\mathcal{ME}$), with the "free will" property necessary for our default reasoning approach to the qualification problem.

Unexpected Observations: Two Light Bulbs Example



- *PressDownSwitch* has action precondition $\neg$*SwitchStuck*
- *PressDownSwitch* causes *LightA* if $\neg$*FaultyBulbA* (i.e. $\neg$*FaultyBulbA* is a fluent precondition). Similarly for *LightB*.
- Normally (i.e. by default) $\neg$*SwitchStuck*, $\neg$*FaultyBulbA*, $\neg$*FaultyBulbB*.
- *PressDownSwitch* is executed, but afterwards $\neg$*LightA* is observed.
- We would like to conclude (*FaultyBulbA* $\vee$ *SwitchStuck*). (Thielscher: solve “weak” and “strong” qualification problems.)

Fluent / Situation Calculus Style Solution?



$Poss(PressDownSwitch, s) \equiv \neg Holds(Ab(SwitchStuck), s)$.

$Poss(PressDownSwitch, s) \rightarrow$
$[Holds(f, Do(PressDownSwitch, s)) \equiv$
$(Holds(f, s) \vee$
$(f = LightA \wedge \neg Holds(Ab(FaultyBulbA), s)) \vee$ ... etc. $)]$.

$\neg Holds(LightA, Do(PressDownSwitch, S0))$.

- Frame problem: given $Holds(CupOnTable, S0)$, we can't conclude $Holds(CupOnTable, Do(PressDownSwitch, S0))$.

$\mathcal{M}$odular-$\mathcal{E}$

- Regards action execution statements as “attempts” to perform actions, which fail (have no effect) if action preconditions are not met.
- Solution to frame problem covers both failed and successful actions.
- Solution to ramification problem covers concurrent actions, non-determinism and looping indirect effects (oscillations). Resolves conflicting effects by non-determinism, taking “race conditions” into account.
- Highly modular and elaboration tolerant. Effect laws can “pick up” extra qualifications from other propositions (“global qualifications”). Any effect law can be consistently added to any consistent theory.
- Enjoys a “free will” property – any action can be attempted in any circumstance (without constraining the past).
- Separates task of qualifying effect laws into “endogenous” and “exogenous” problems, with the latter solvable by simple minimization.

$\mathcal{M}$odular-$\mathcal{E}$ Syntax

h-propositions of the form: ϕ `holds-at` T
o-propositions of the form: A `occurs-at` T
c-propositions of the form: C `causes` L
p-propositions of the form: ϕ `prevents` E
a-propositions of the form: `always` ϕ
n-propositions of the form: `normally` L

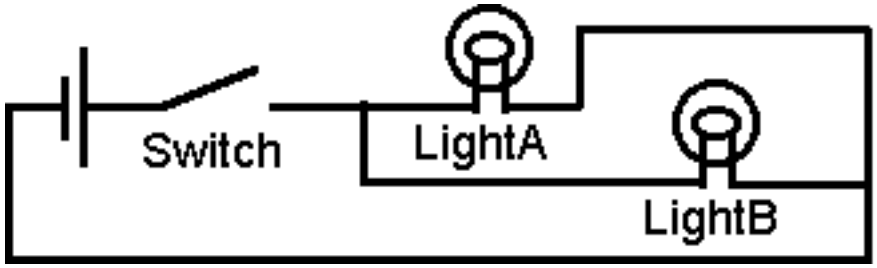
(ϕ = fluent formula, L = fluent literal, C = set of fluent and action literals, E = set of action constants and fluent literals.)

C `causes` L – "C provisionally causes L" (qualified both locally by C and globally via other c-, p- and a-props).

ϕ `prevents` E – "ϕ prevents the actions and fluents in E from being executed/caused simultaneously."

`always` ϕ – "$\neg\phi$ is unobservable."

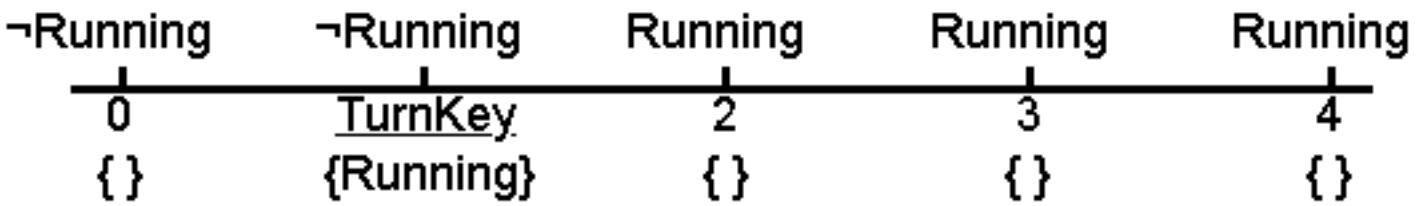
Two Light Bulbs Example in $\mathcal{ME}$



SwitchStuck `prevents` *PressDownSwitch*
PressDownSwitch `causes` *ElectricCurrent*
$\{$*ElectricCurrent*, $\neg$*FaultyBulbA*$\}$ `causes` *LightA*
$\{$*ElectricCurrent*, $\neg$*FaultyBulbB*$\}$ `causes` *LightB*
`always` ($\neg$*ElectricCurrent* $\rightarrow$ ($\neg$*LightA* $\wedge$ $\neg$*LightB*))
`normally` $\neg$*SwitchStuck*
`normally` $\neg$*FaultyBulbA*
`normally` $\neg$*FaultyBulbB*
PressDownSwitch `occurs-at 1`
$\neg$*LightA* `holds-at 2`

$\mathcal{M}$odular-$\mathcal{E}$ Semantics

- **Interpretations** are mappings from fluent/timepoint pairs to $\{true, false\}$.
- **Models** are interpretations in which every fluent literal persists until its converse appears in a timepoint's **change set**.



- Change sets are computed by identifying **proper causal descendants** within **causal chains**.
- In a causal chain, causal laws trigger (instantaneous) **processes**, which are then **resolved** further along the chain.

$$\langle \{\neg Running\}, \{TurnKey\}, \emptyset \rangle \rightarrow \langle \{\neg Running\}, \{TurnKey\}, \{\uparrow Running\} \rangle \rightarrow \langle \{Running\}, \emptyset, \emptyset \rangle$$

The Role of P-propositions

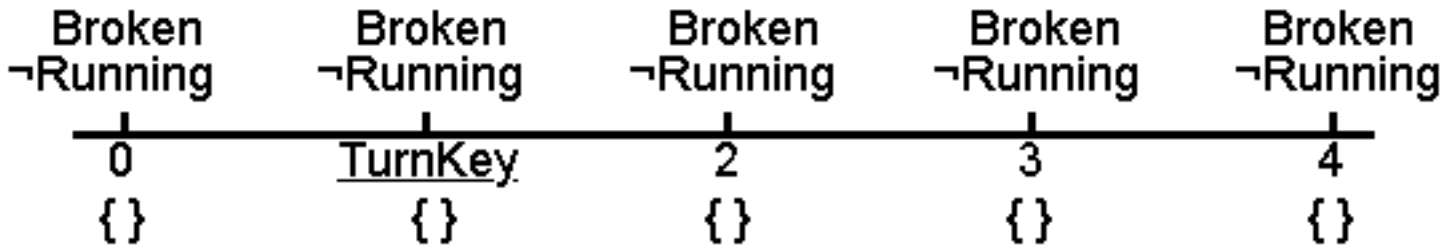
P-propositions prevent c-propositions from triggering processes within causal chains:

TurnKey `causes` *Running*

TurnKey `occurs-at 1`

Broken `prevents` *Running*

$$\langle \{Broken, \neg Running\}, \{TurnKey\}, \emptyset \rangle$$
$$\downarrow$$
$$\langle \{Broken, \neg Running\}, \{TurnKey\}, \emptyset \rangle$$
$$\downarrow$$
$$\underline{\langle \{Broken, \neg Running\}, \emptyset, \emptyset \rangle}$$

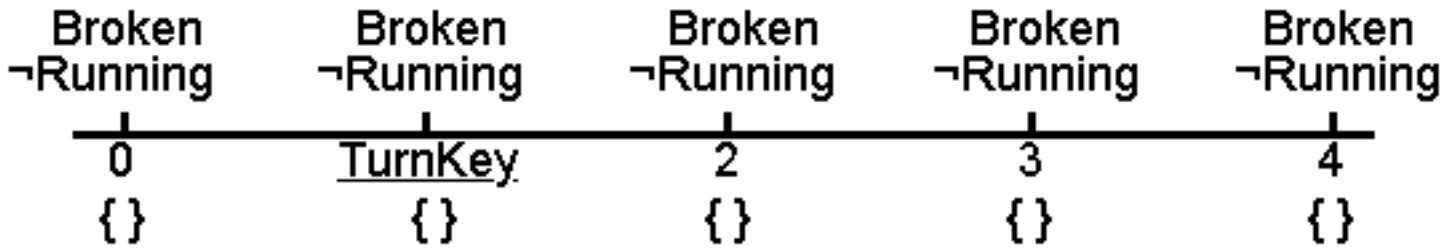


The Role of A-propositions

A-propositions provide (“global”) qualifications to effect laws by constraining the selection of proper causal descendants within causal chains:

TurnKey `causes` *Running*
TurnKey `occurs-at 1`
`always` $\neg(Broken \wedge Running)$

$$\underline{\langle\{Broken, \neg Running\}, \{TurnKey\}, \emptyset\rangle}$$
$$\downarrow$$
$$\langle\{Broken, \neg Running\}, \{TurnKey\}, \{\uparrow Running\}\rangle$$
$$\downarrow$$
$$\langle\{Broken, Running\}, \emptyset, \emptyset\rangle$$



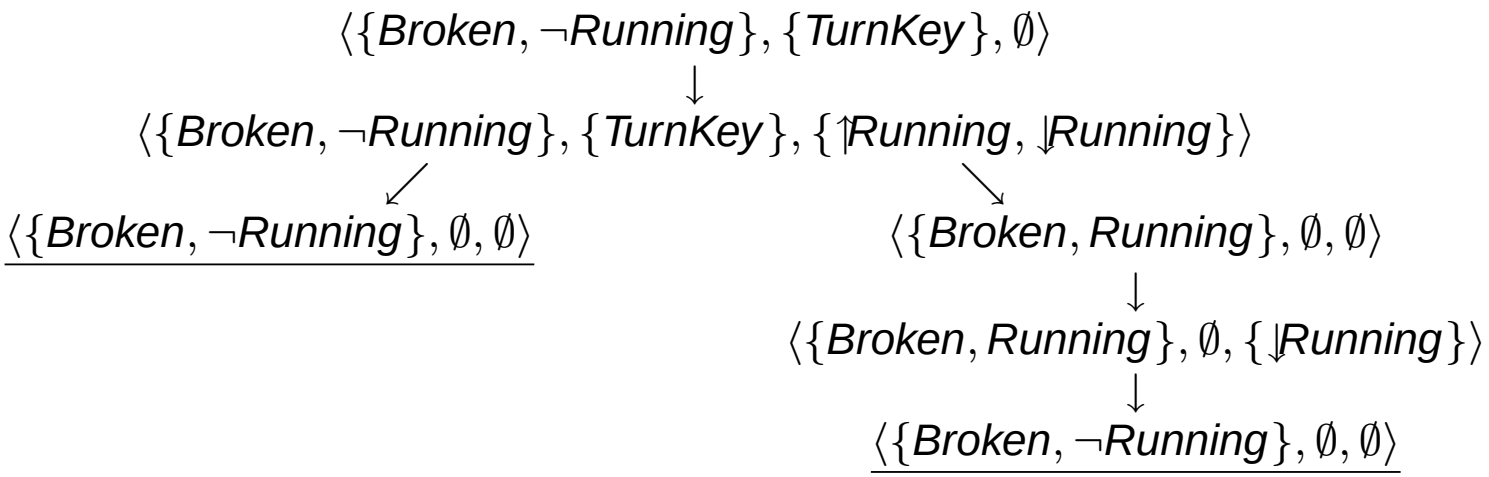
Interaction Between C-propositions

Conflicting c-propositions result in conflicting processes, which resolve non-deterministically:

TurnKey `causes` *Running*

TurnKey `occurs-at 1`

Broken `causes` ¬*Running*



Modularity, Elaboration Tolerance and Free Will Results

- **Free Will Theorem:** Let M be a model of a finite domain description D that doesn't include observations after timepoint T. Then for any action A there is a model of $D \cup \{A \texttt{ occurs-at } T\}$ identical to M up until T.

- **Causal Elaboration Tolerance Theorem:** Let D_a be a consistent collection of a-propositions and let E be a finite set of o-, c- and p-propositions. Then $D_a \cup E$ is consistent.

- In $\mathcal{M}$odular-$\mathcal{E}$, given a consistent initial situation, **inconsistency arises only when observations conflict with predictions** (generated from earlier observations together with causal laws).

- Allows a clean separation of "endogenous" and "exogenous" qualification problems – the latter now solvable by **simple minimization**.

$\mathcal{ME}$'s Approach to Exogenous Qualification

- Transform domain D into the **default domain description** D_d:
  - "C `causes` L" becomes "$C \cup \{$*NormExo*$($*unique-id*$)\}$ `causes` L"
  - for each action A add "$\neg$*NormExo*(A) `prevents` A"
  - add " `normally` *NormExo*$(...)$" for each *NormExo*$(...)$
- **Default models** of D are defined as maximal (w.r.t. *NormExo* fluents) models of D_d.
- If there are no observations after an initial time-point, this transformation has no effect.

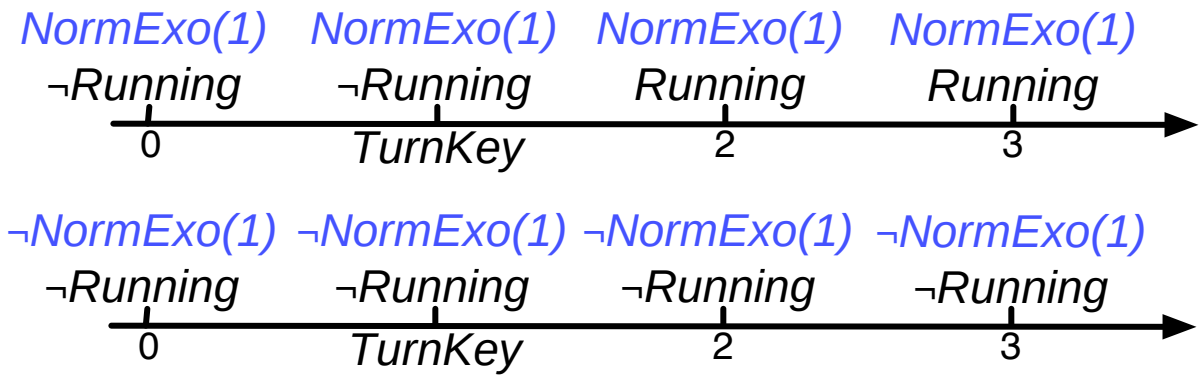
Exogenous "Failures" Example

TurnKey causes *Running* (1)

TurnKey occurs-at 1 (2)

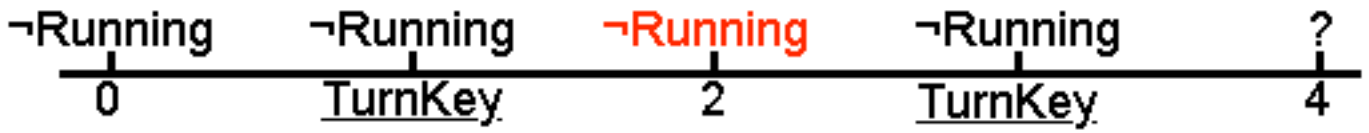
¬*Running* holds-at 2 (3)

- Replace (1) by: {*NormExo*(1), *TurnKey*} causes *Running*
  and add: normally *NormExo*(1)



Exogenous “Failures” & Exogenous “Accidents”

TurnKey causes *Running* (1)
TurnKey occurs-at {1, 3} (2)
¬*Running* holds-at 2 (3)



- **Failures only**: “¬*Running* holds-at 4” given by replacing (1) by:

  {*NormExo*(1), *TurnKey*} causes *Running*
  normally *NormExo*(1)

- **Accidents only**: “*Running* holds-at 4” given by replacing (1) by:

  {¬*AbnormExoAction*(1), *TurnKey*} causes *Running*

- **Recovery policies** can mix and match as required.

Conclusions

- **Narrative-based formalisms**, with explicit notions of "**observation**" and "**action occurrence**", are preferable for specifying (aspects of) the qualification problem.
- **Modular** and **elaboration tolerant** frameworks with characteristics such as the "**free will**" property facilitate clean solutions to the qualification problem, and help disambiguate its various aspects.
- Complete and elaboration tolerant solutions to the ramification and "endogenous" qualification problems enable the **"exogenous" qualification problem** to be tackled using **straightforward minimization**.
- The notion of (micro) **processes** is useful to avoid over-constraining models in domains involving concurrency, non-determinism, **oscillations** and/or "**race conditions**" between competing effects.

Ideas for Future Work

- Study "recovery policies" and how to succinctly specify them.
- Improve the implemention of (a useful fragment of) $\mathcal{M}$odular-$\mathcal{E}$. (Show soundness/completeness?)
- Look at the computational qualification problem (how can reasoning disregard "abnormal" conditions unless there is evidence for them?).
- Study further the differences between various types of causal and non-causal relationships that might hold between fluents (relationships at different temporal granularities, "definitional" relationships, etc.).
- Export some of $\mathcal{M}$odular-$\mathcal{E}$'s notions to classical formalisms such as the Event Calculus.
- Do some empirical testing to see the extent to which $\mathcal{M}$odular-$\mathcal{E}$ corresponds to common sense.
- Applications: Computational biology? $\mathcal{M}$odular-$\mathcal{E}$ in control modules?

Extra Slides ...

Faulty Circuit Example

SwitchOn causes *ElectricCurrent*

ElectricCurrent causes *BrokenFuse*

BrokenFuse causes $\neg$*ElectricCurrent*

always $\neg$(*ElectricCurrent* $\wedge$ *BrokenFuse*)

$$\langle \{\neg \mathit{ElectricCurrent}, \neg \mathit{BrokenFuse}\}, \{\mathit{SwitchOn}\}, \emptyset \rangle$$
$$\downarrow$$
$$\langle \{\neg \mathit{ElectricCurrent}, \neg \mathit{BrokenFuse}\}, \{\mathit{SwitchOn}\}, \{\uparrow \mathit{ElectricCurrent}\} \rangle$$
$$\downarrow$$
$$\langle \{\mathit{ElectricCurrent}, \neg \mathit{BrokenFuse}\}, \emptyset, \emptyset \rangle$$
$$\downarrow$$
$$\langle \{\mathit{ElectricCurrent}, \neg \mathit{BrokenFuse}\}, \emptyset, \{\uparrow \mathit{BrokenFuse}\} \rangle,$$
$$\downarrow$$
$$\langle \{\mathit{ElectricCurrent}, \mathit{BrokenFuse}\}, \emptyset, \emptyset \rangle$$
$$\downarrow$$
$$\langle \{\mathit{ElectricCurrent}, \mathit{BrokenFuse}\}, \emptyset, \{\downarrow \mathit{ElectricCurrent}\} \rangle$$
$$\downarrow$$
$$\underline{\langle \{\neg \mathit{ElectricCurrent}, \mathit{BrokenFuse}\}, \emptyset, \emptyset \rangle}$$

Non-deterministic Lift Door Example

$\{PressOpen\}$ `causes` $DoorOpen$

$\{PressClose\}$ `causes` $\neg DoorOpen$

$DoorOpen$ `holds-at` `1`

$PressOpen$ `occurs-at` `2`

$PressClose$ `occurs-at` `2`

$$\langle \{DoorOpen\}, \{PressClose, PressOpen\}, \emptyset \rangle$$

$$\downarrow$$

$$\langle \{DoorOpen\}, \{PressClose, PressOpen\}, \{\uparrow DoorOpen, \downarrow DoorOpen\} \rangle$$

$$\swarrow \qquad \searrow$$

$$\underline{\langle \{DoorOpen\}, \emptyset, \emptyset \rangle} \qquad \underline{\langle \{\neg DoorOpen\}, \emptyset, \emptyset \rangle}$$

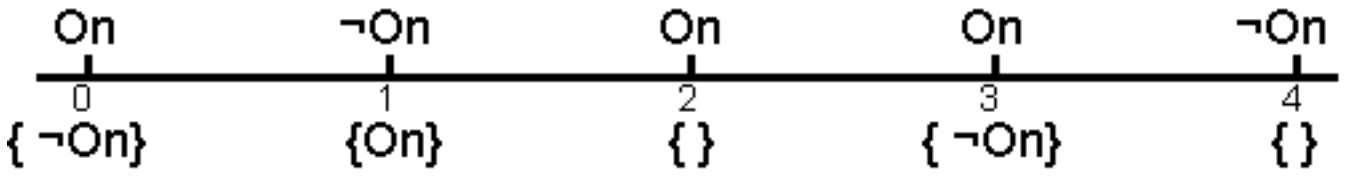
An Oscillator

$\{On\}$ causes $\neg On$

$\{\neg On\}$ causes On

$$\underline{\langle\{On\},\emptyset,\emptyset\rangle}$$
$$\downarrow$$
$$\langle\{On\},\emptyset,\{\downarrow On\}\rangle$$
$$\downarrow$$
$$\underline{\langle\{\neg On\},\emptyset,\emptyset\rangle}$$
$$\downarrow$$
$$\langle\{\neg On\},\emptyset,\{\uparrow On\}\rangle$$
$$\downarrow$$
$$\underline{\langle\{On\},\emptyset,\emptyset\rangle}$$

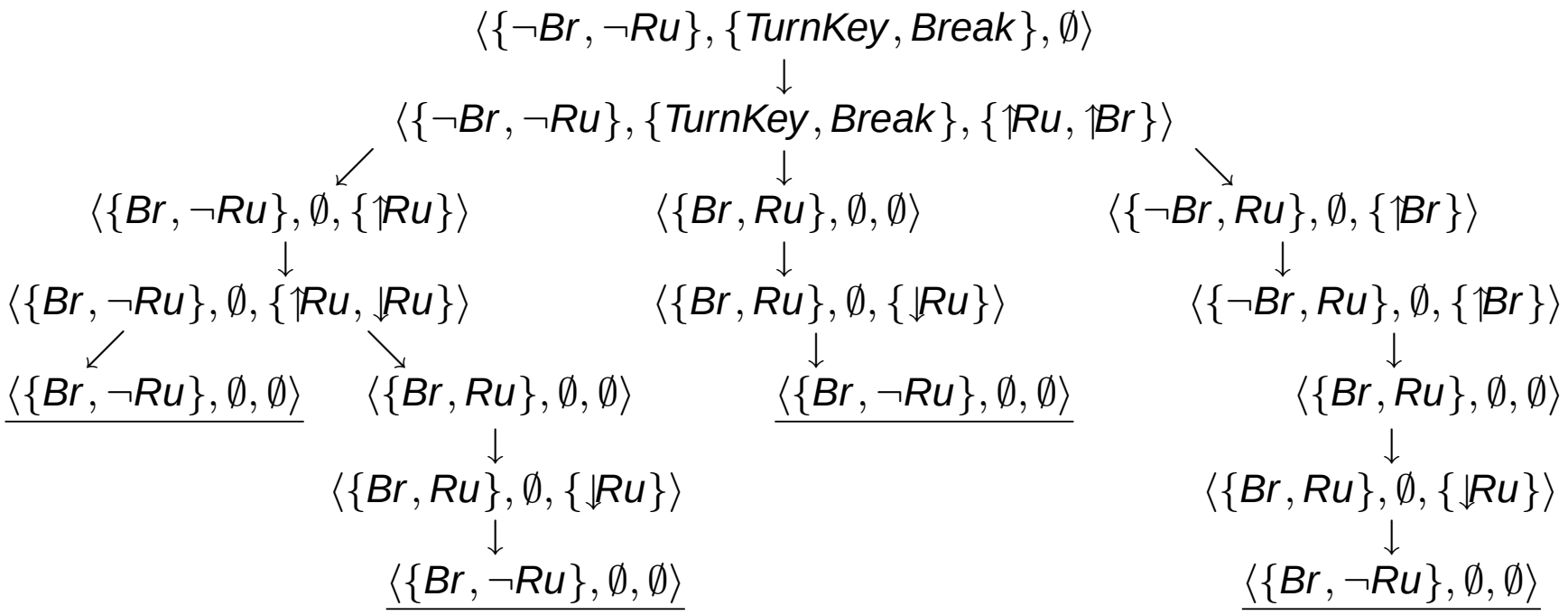
$$\underline{\langle\{\neg On\},\emptyset,\emptyset\rangle}$$
$$\downarrow$$
$$\langle\{\neg On\},\emptyset,\{\uparrow On\}\rangle$$
$$\downarrow$$
$$\underline{\langle\{On\},\emptyset,\emptyset\rangle}$$
$$\downarrow$$
$$\langle\{On\},\emptyset,\{\downarrow On\}\rangle$$
$$\downarrow$$
$$\underline{\langle\{\neg On\},\emptyset,\emptyset\rangle}$$



Broken Car Example C

TurnKey causes *Running*
always $\neg(Broken \wedge Running)$
TurnKey occurs-at 1

Break causes *Broken*
Broken causes $\neg Running$
Break occurs-at 1

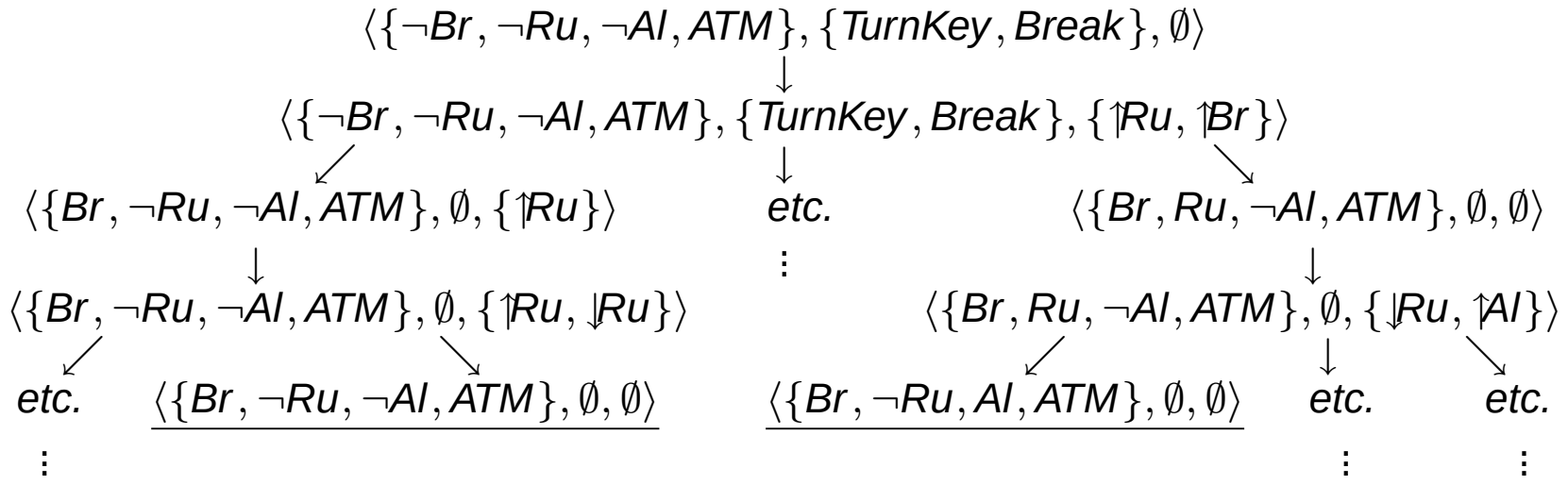


Broken Car Example D, with “Race Conditions”

When the car is left in anti-theft mode (ATM), running the engine (even for an instant) will trigger the alarm:

TurnKey causes *Running*
always $\neg(Broken \wedge Running)$
$\{Running, ATM\}$ causes *Alarm*

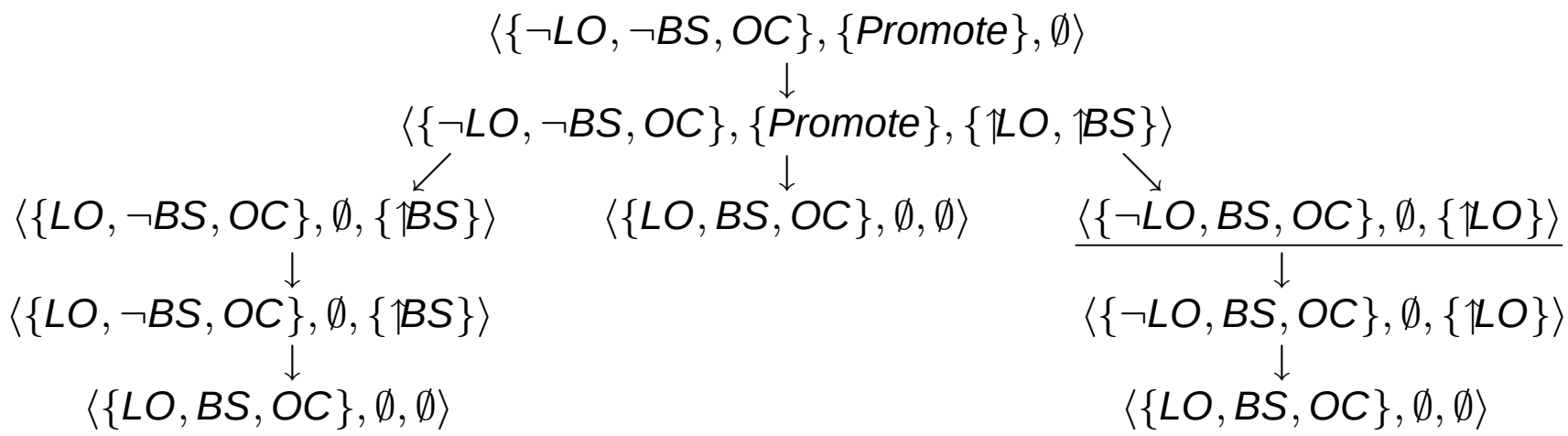
Break causes *Broken*
Broken causes $\neg$*Running*
$\{Break, TurnKey\}$ occurs-at 1



Promotion Example

An employee gets promoted at time 1. Promotion results in a large office (LO) and big salary (BS). But nobody gets a large office when the building is overcrowded (OC), which it is at time 1:

$$\textit{Promote} \texttt{ causes } \{BS, LO\} \qquad (\neg LO \wedge \neg BS \wedge OC) \texttt{ holds-at } 1$$
$$\texttt{always } \neg(OC \wedge LO) \qquad \textit{Promote} \texttt{ occurs-at } 1$$



Definition of Triggering

Let D be a domain description, $N = \langle S, B, P \rangle$ a node, L_t a set of fluent literals, $P_t = \{\text{proc}(L) \mid L \in L_t\}$, and B_t a set of action constants. The set $(B_t \cup P_t)$ is **triggered at** N **with respect to** D iff

1. $B_t \subseteq B$
2. For each p-proposition "ϕ `prevents` E" in D, either ϕ is not satisfied in S or $E \not\subseteq (B_t \cup L_t)$.
3. For each $L \in L_t$ there is a c-proposition "C `causes` L" in D such that (i) for each action constant $A \in C$, $A \in B_t$, (ii) for each action literal $\neg A \in C$, $A \notin B_t$, and (iii) for each fluent literal $L' \in C$, $L' \in S$.

$(B_t \cup P_t)$ is **maximally triggered at** N **with respect to** D iff there is no other set $(B'_t \cup P'_t)$ also triggered at N with respect to D and $(B_t \cup P_t)$ is a strict subset of $(B'_t \cup P'_t)$.

Stationary And Static Nodes

Let D be a domain description and $N = \langle S, B, P \rangle$ a causal node. N is **stationary** iff for each resolvant $\langle S', \emptyset, P' \rangle$ of N, $S' = S$. N is **static w.r.t.** D iff every process successor of N w.r.t. D is stationary.

Causal Chain Definition

Let D be a domain description and let N_0 be a node. A **causal chain rooted at N_0 with respect to D** is a (finite) sequence $N_0, N_1, ..., N_{2n}$ of nodes such that for each k, $0 \leq k \leq n-1$, N_{2k+1} is a process successor of N_{2k} w.r.t. D and N_{2k+2} is a resolvant of N_{2k+1}, and such that the following conditions hold:

1. N_{2n} is fully resolved.
2. N_{2n} is static, or there exists $k < n$ s.t. $N_{2n} = N_{2k}$.
3. If there exists $j < k \leq n$ s.t. $N_{2j} = N_{2k}$ then $k = n$.
4. There does not exist a $k < n$ s.t. N_{2k} is static.

Proper Causal Descendant Definition

Let D be a domain description and let N_0 and N be nodes. N is a **proper causal descendant of** N_0 **w.r.t.** D iff N is a-consistent w.r.t. D, and there exists a causal chain $N_0, N_1, ..., N_{2n}$ w.r.t. D such that $N = N_{2k}$ for some $0 \leq k \leq n$ and at least one of the following two conditions holds:

1. There exists $j \leq k$ such that $N_{2j} = N_{2n}$.
2. There does not exist a causal chain $N_0, N_1, ..., N_{2k}, N'_{2k+1}, ..., N'_{2m}$ w.r.t. D and a j such that $k < j \leq m$ and N'_{2j} is a-consistent w.r.t. D.

A state S is **stable w.r.t.** D if there exists a node $\langle S, \emptyset, P \rangle$ which is a proper causal descendant of $\langle S, \emptyset, \emptyset \rangle$.

Definition of a Model

Let D be a domain description, and let Φ^* be the set of all (+ve and -ve) fluent literals in the language. Then an interpretation H is a **model** of D iff there exists a mapping $c : \Pi \mapsto 2^{\Phi^*}$ such that for all T, $c(T)$ is a change set at T w.r.t. H, and the following three conditions hold. For every fluent literal L and time-points $T_1 \prec T_3$:

1. If H satisfies L at T_1, and there is no time-point T_2 s.t. $T_1 \preceq T_2 \prec T_3$ and $\overline{L} \in c(T_2)$, then H satisfies L at T_3.

2. If $L \in c(T_1)$, and there is no time-point T_2 such that $T_1 \prec T_2 \prec T_3$ and $\overline{L} \in c(T_2)$, then H satisfies L at T_3.

3. H satisfies the following constraints:
   - For all "ϕ `holds-at` T" in D, H satisfies ϕ at T.
   - For all time-points T, $S(H, T)$ is a stable state.

Abductive and Deductive Planning

In the Event Calculus

"$\mathit{Happens}(\mathit{TurnKey}, 1)$" is a plan for "$\mathit{Holds}(\mathit{HasKey}, 2)$"

can be expressed abductively as:

$$\mathit{THEORY} \wedge [\mathit{Happens}(a, t) \equiv (a = \mathit{TurnKey} \wedge t = 1)] \models \mathit{Holds}(\mathit{HasKey}, 2)$$

or deductively as:

$$\mathit{THEORY} \models [\mathit{Happens}(a, t) \equiv (a = \mathit{TurnKey} \wedge t = 1)] \rightarrow \mathit{Holds}(\mathit{HasKey}, 2)$$

In general, the Deduction Theorem ensures that:

$$(\mathit{THEORY} \wedge \mathit{PLAN}) \models \mathit{GOAL} \quad \text{iff} \quad \mathit{THEORY} \models (\mathit{PLAN} \rightarrow \mathit{GOAL})$$

Anomalous Plans: Event Calculus Example

- Domain description: $\forall t[\mathit{Poss}(\mathit{TurnKey}, t) \equiv \mathit{Holds}(\mathit{HasKey}, t)]$
- EC general axioms including: $\forall a, t[\neg \mathit{Poss}(a, t) \rightarrow \neg \mathit{Happens}(a, t)]$
- EC Abductive Planning: Find *PLAN* s.t. $(\mathit{THEORY} \wedge \mathit{PLAN}) \models \mathit{GOAL}$
- (Completion of) "$\mathit{Happens}(\mathit{TurnKey}, 1)$" is a plan for "$\mathit{Holds}(\mathit{HasKey}, 2)$"! (And worse - it's a plan for "$\mathit{Holds}(\mathit{HasKey}, 0)$".)
- Suspect similar problems for any formalism (e.g. TAL, Language C) that represents action preconditions via narrative constraints, e.g:

$$\bot \Leftarrow i : (\mathit{PushBox}(l) \wedge \mathit{Loc}(\mathit{Monkey}) \neq \mathit{Loc}(\mathit{Box}))$$

Anomalous Plans: Language $\mathcal{E}$ Example

TurnKey `initiates` *Running*

$\neg$*Running* `whenever` $\{$*Broken*$\}$

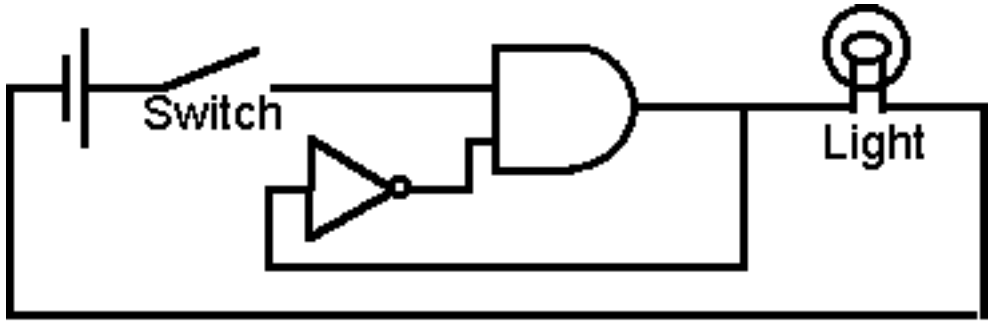
3 models:
(a) $\forall t$: $\langle$*Running*, $\neg$*Broken*$\rangle$
(b) $\forall t$: $\langle\neg$*Running*, $\neg$*Broken*$\rangle$
(c) $\forall t$: $\langle\neg$*Running*, *Broken*$\rangle$

Addition of "*TurnKey* `happens-at 1`" gives instead 2 models:

(a) $\forall t$: $\langle$*Running*, $\neg$*Broken*$\rangle$
(b′) $\forall t \leq 1$: $\langle\neg$*Running*, $\neg$*Broken*$\rangle$
$\forall t > 1$: $\langle$*Running*, $\neg$*Broken*$\rangle$

so is a plan for "$\neg$*Broken* `holds-at 2`".

Anomalous Plans: Language $\mathcal{E}$ Example 2



PressDownSwitch `initiates` *SwitchConnected* `when` $\{\neg$*SwitchStuck*$\}$
$\neg$*Light* `whenever` $\{\neg$*SwitchConnected*$\}$
$\neg$*Light* `whenever` $\{$*SwitchConnected*, *Light*$\}$
Light `whenever` $\{$*SwitchConnected*, $\neg$*Light*$\}$

2 models: (a) $\forall t : \langle \neg$*SwitchStuck*, $\neg$*SwitchConnected*, $\neg$*Light*$\rangle$
(b) $\forall t : \langle$*SwitchStuck*, $\neg$*SwitchConnected*, $\neg$*Light*$\rangle$

Addition of "*PressDownSwitch* `happens-at 1`" eliminates model (a), so is a plan for "*SwitchStuck* `holds-at 2`".

Extra/Modified Frame Axioms?

- For the Situation Calculus:

$$\neg \mathit{Poss}(a, s) \rightarrow [\mathit{Holds}(f, \mathit{Do}(a, s)) \equiv \mathit{Holds}(f, s)]$$

- For the Event Calculus:

$$\mathit{HoldsAt}(f, t_2) \leftarrow [\mathit{HoldsAt}(f, t_1) \wedge t_1 < t_2 \wedge \neg \mathit{Clipped}(t_1, f, t_2)]$$

$$\mathit{Clipped}(t_1, f, t_2) \equiv \exists a, t[\mathit{Happens}(a, t) \wedge t_1 \leq t < t_2 \wedge \mathit{Poss}(a, t) \wedge \mathit{Terminates}(a, f, t)]$$

$$\mathit{HoldsAt}(f, t_2) \leftarrow [\mathit{Happens}(a, t_1) \wedge \mathit{Initiates}(a, f, t_1) \wedge \mathit{Poss}(a, t_1) \wedge t_1 < t_2 \wedge \neg \mathit{Clipped}(t_1, f, t_2)]$$

Some References

- A. Kakas, L. Michael and R. Miller. *Modular-E: an Elaboration Tolerant Approach to the Ramification and Qualification Problems*. Proceedings of LPNMR 2005. (Journal version to appear in AIJ.)

- R. Miller and M. Shanahan. *Some Alternative Versions of the Event Calculus*. Lecture Notes in Artificial Intelligence, vol. 2408, 2002.

- R. Miller. *Three Problems in Logic-based Knowledge Representation*. ASLIB Proceedings: New Information Perspectives, vol. 58, 2006.

- M. Thielscher. *The Qualification Problem: A Solution to the Problem of Anomalous Models*. AIJ, 131(1-2):1-37, 2001.

- A. Herzig and I. Varzinczak. *Domain Descriptions Should be Modular*. Proceedings of ECAI 2004.

Some Existing Applications of the Event Calculus

- planning
- cognitive robotics
- abductive reasoning
- database updates
- accident report processing
- legal reasoning
- modelling continuous change and mathematical modelling
- modelling and reasoning about agent beliefs
- reasoning about programming constructs
- and software engineering