Pertinence for Causal Representation of Action Domains

Pedro Cabalar

PhD dissertation
A Coruña, 2001

Outline

☞ **Introduction**

❑ Pertinence Logic L^2

❑ Action domains and transition systems

❑ $\mathcal{P}$-language: operational semantics

❑ Logical formalization

❑ Causal cycles

❑ Related work

❑ System PAL

❑ Conclusions

Scope

- Artificial Intelligence

- Reasoning about Actions and Change
  Programs with commonsense [McCarthy59]

- Logical representation of dynamic domains

- Key topics: inertia and causality.

Causality

A motivating example:

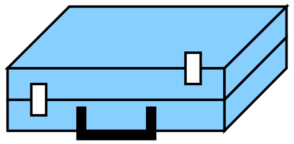
$$f = m \cdot a$$

$$\begin{aligned} a &= f/m \quad \text{(prediction)} \\ m &= f/a \quad \text{(explanation)} \\ f &= m \cdot a \quad \text{(planning)} \end{aligned}$$

m is not an effect of f and a.

Causality

Lin's suitcase.



If both locks are lifted up, the suitcase gets open

$$up(1) \wedge up(2) \supset open \tag{1}$$

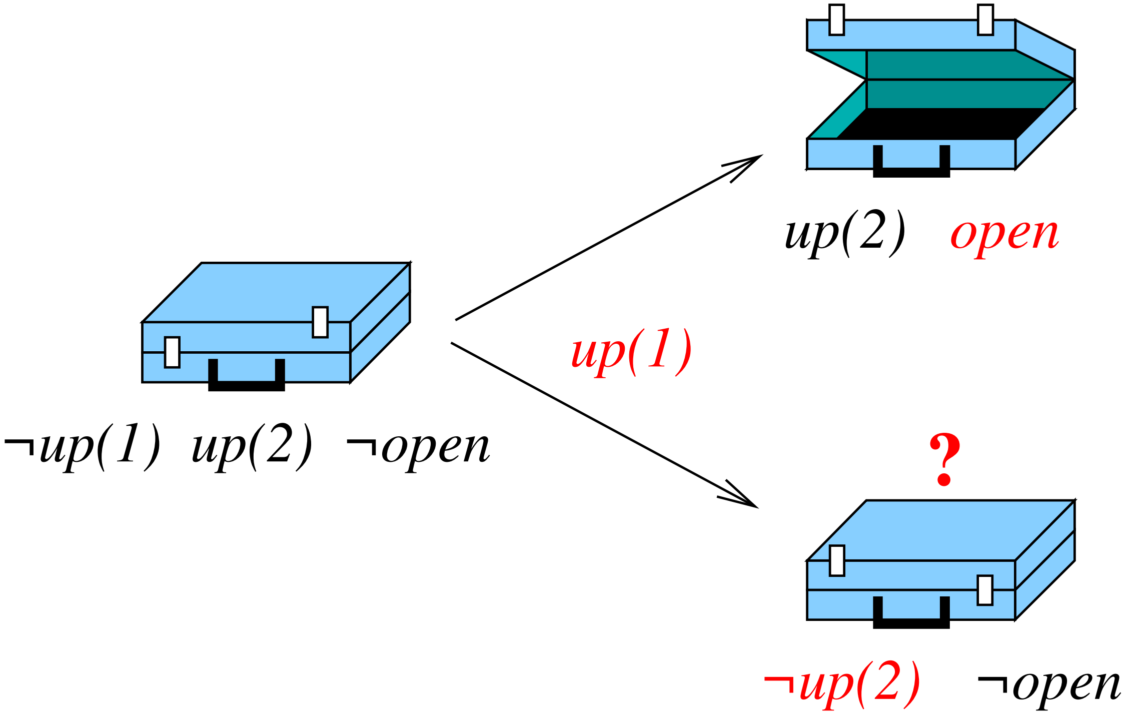
(prediction) . . . but it can also be "used" as:

$$up(1) \wedge \neg open \supset \neg up(2)$$

(explanation)

Problems when considering change

Example: change to a new state with $up(1)$ true (we lift lock 1), satisfying $up(1) \wedge up(2) \supset open$ but minimizing changes (inertia) ...



Solution: replace material implication by a causal conditional.

Causal approaches: two orientations

1. *Inferential* causality: causal conditionals $\approx$ inference rules.

2. *Change-based* causality:
   material implication $+$ a special predicate

   - "persisted" vs. "caused"
   - "caused" is auxiliary: only for ruling out undesired models

Our proposal: using both but providing ...

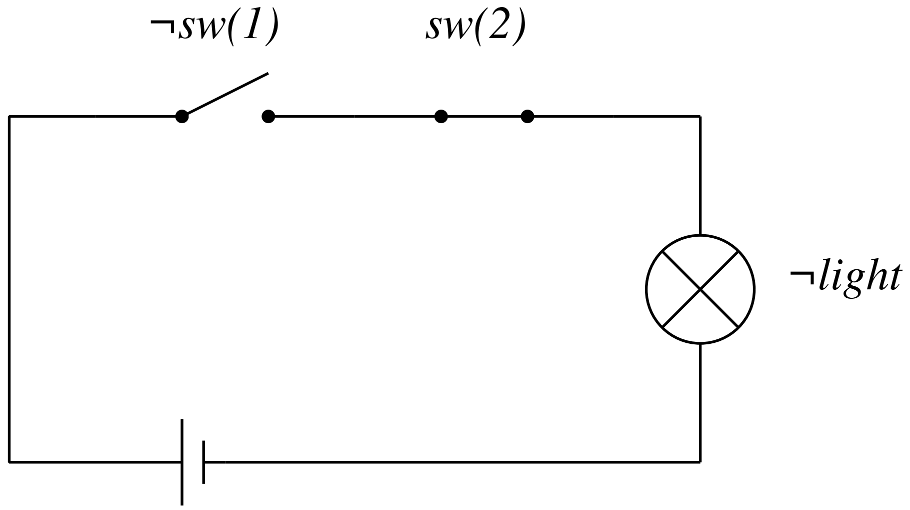
A meaning for “caused”

- Intuition: a fact is *caused* when it is obtained as a direct or indirect result of the performed actions.

  . . . otherwise, it simply persists.

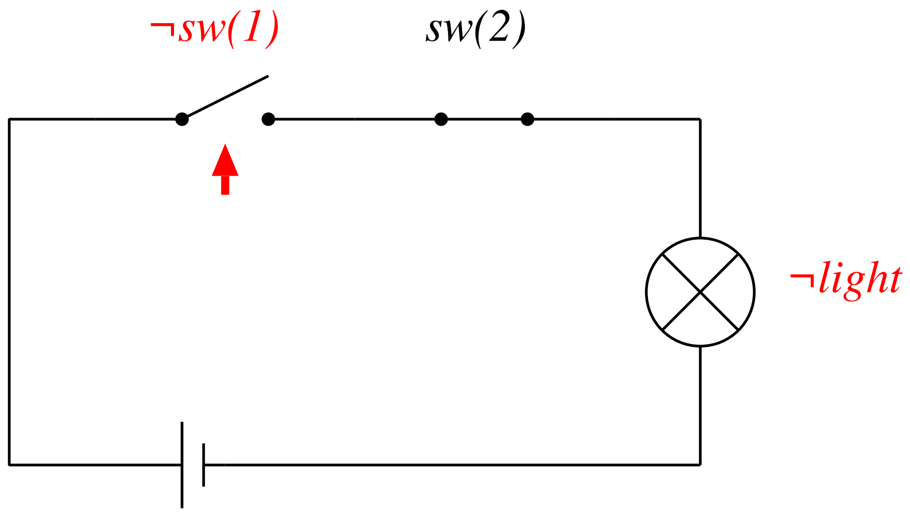
- Caused means “relevantly affected by the performed actions” : we will call it pertinent for short.

Example



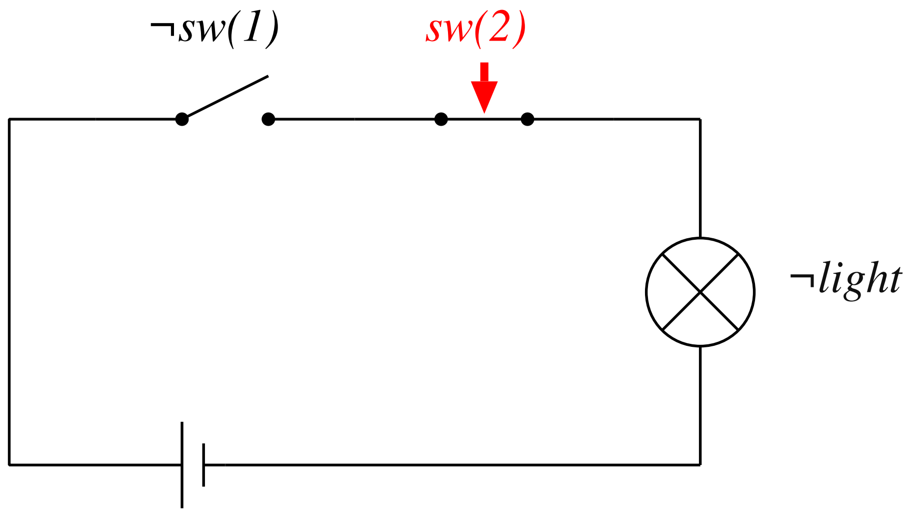
How could state $\neg sw(1), sw(2), \neg light$ have been obtained?

Example



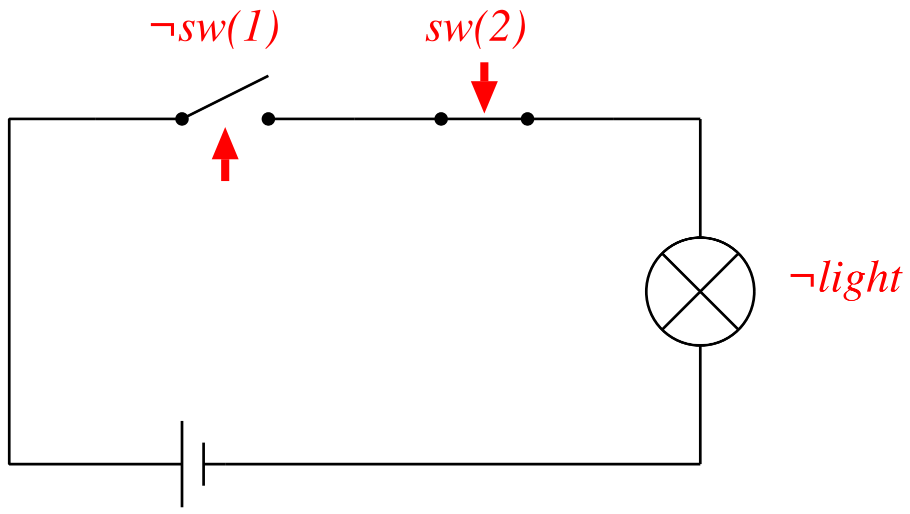
Opening $sw(1)$ while $sw(2)$ was closed $\Rightarrow$ $\neg light$ caused

Example



closing $sw(2)$ while $sw(1)$ was open $\Rightarrow$ $\neg light$ persisted

Example



Both opening $sw(1)$ and closing $sw(2)$ $\Rightarrow$ $\neg light$ caused

Intuitive postulates

P1) A fact is caused (or *pertinent*) whenever it is a direct or indirect effect of the performed actions.

P2) Any fact is obtained either by causal chaining or by inertia, but not both.

P3) A change of fluent value can only occur due to causation (*pertinence*).

P4) "Caused" does not necessarily mean a change of value.

P5) Conditions of causal rules are applicable iff they hold and are pertinent.

Outline

- ✔ Introduction
- ☞ **Pertinence Logic L^2**
- ❑ Action domains and transition systems
- ❑ $\mathcal{P}$-language: operational semantics
- ❑ Logical formalization
- ❑ Causal cycles
- ❑ Related work
- ❑ System PAL
- ❑ Conclusions

Pertinence Logic L^2 [Otero97]

- Static and propositional. Finite signature $= \Sigma$.

- Syntax: propositional logic plus $\phi \Leftarrow \psi$

- Semantics: double valuation (truth and pertinence).
  Interesting: formulas can be pertinent.

- Interpretation, M is a pair $M^t M^p$ of functions s.t.
  for any $a \in \Sigma$, $M^t(a) \in \{\mathtt{t}, \mathtt{f}\}$ and $M^p(a) \in \{\mathtt{p}, \mathtt{n}\}$.

  For short, four values: $M(a) \in \{\mathtt{tp}, \mathtt{tn}, \mathtt{fp}, \mathtt{fn}\}$.

L^2 semantics

- Valuation of a formula α without $\Leftarrow$
  - $M^t(\alpha)$ is the standard truth-valuation.
  - $M^p(\alpha) = \mathtt{p}$ iff $a \in M^p$ for some atom a in ϕ.

- Example: $\Sigma = \{a, b, c, d, e\}$ $M_0^t = \{a, b, e\}$ $M_0^p = \{b, c, e\}$.

$$\begin{array}{cccccc} \neg & (& a & \vee & c &) \\ \mathtt{f} & & \mathtt{t} & \mathtt{t} & \mathtt{f} & \end{array} \qquad \mathtt{fp}$$

$$\begin{array}{cccc} a & \wedge & \neg & d \\ \mathtt{t} & \mathtt{t} & \mathtt{t} & \mathtt{f} \end{array} \qquad \mathtt{tn}$$

L^2 semantics

- Valuation of $\phi \Leftarrow \psi$
  - $M^t(\phi \Leftarrow \psi) = \mathtt{t}$ iff $M(\psi) \neq \mathtt{tp}$ or $M(\phi) = \mathtt{tp}$,
  - $M^p(\phi \Leftarrow \psi) = \mathtt{p}$ iff $M(\psi) = \mathtt{tp}$.

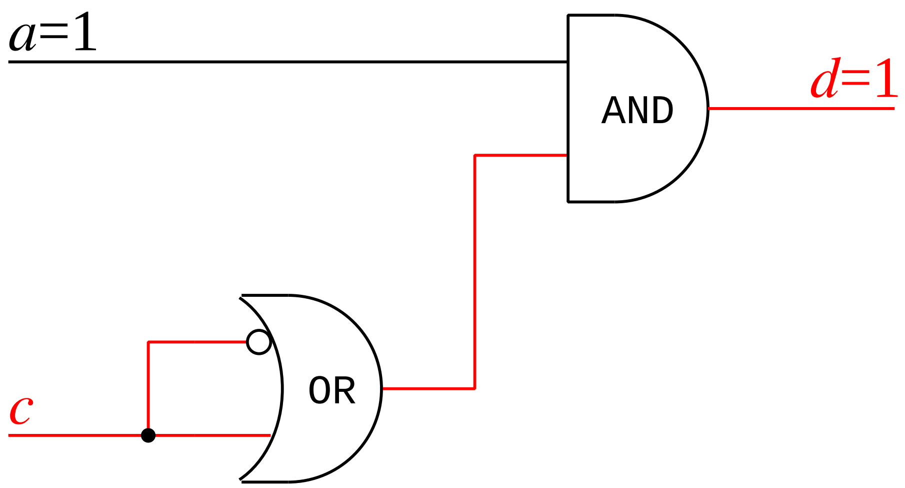
- Example: $M_0^t = \{a, b, e\}$ $M_0^p = \{b, c, e\}$.

$$\begin{array}{ccc} & \mathtt{t} & \\ e & \Leftarrow & b \\ \mathtt{tp} & & \mathtt{tp} \end{array} \qquad \begin{array}{ccc} & \mathtt{t} & \\ d & \Leftarrow & a \\ \mathtt{tn} & & \mathtt{tn} \end{array} \qquad \begin{array}{ccc} & \mathtt{f} & \\ d & \Leftarrow & e \\ \mathtt{tn} & & \mathtt{tp} \end{array}$$

- Model M of a theory T: for all $\phi \in T$, $M^t(\phi) = \mathtt{t}$.
- Truth-equivalence: $M^t(\phi) = M^t(\psi)$.

Relevance of tautologies

$$\begin{array}{ccc} & \mathtt{t} & \\ d & \Leftarrow & a \\ \mathtt{tn} & & \mathtt{tn} \end{array} \qquad\qquad \begin{array}{ccc} & \mathtt{f} & \\ d & \Leftarrow & a \wedge (c \vee \neg c) \\ \mathtt{tn} & & \mathtt{tp} \end{array}$$



- changing c, the persistence of d not guaranteed!

Other operators

$!\phi$	ϕ pertinent
$\mathbb{C}\,\phi \stackrel{\text{def}}{=} \phi \wedge !\phi$	ϕ has been caused
	$\phi \Leftarrow \psi$ is truth-equivalent to $\mathbb{C}\,\psi \supset \mathbb{C}\,\phi$
$\mathbb{P}\,\phi \stackrel{\text{def}}{=} \phi \wedge \neg !\phi$	ϕ has persisted
$\phi \vert \psi$	persistence-cut disjunction
$\phi \& \psi$	persistence-cut conjunction

- a set of transformation rules allows simplifying any L^2 theory into a classical one with atoms like:

$$\begin{array}{rcl} !a & \longrightarrow & pert(a) \\ a & \longrightarrow & holds(a) \end{array}$$

Outline

- ✔ Introduction
- ✔ Pertinence Logic L^2
- ☞ **Action domains and transition systems**
- ❑ $\mathcal{P}$-language: operational semantics
- ❑ Logical formalization
- ❑ Causal cycles
- ❑ Related work
- ❑ System PAL
- ❑ Conclusions

Action domains

- fluents: $\mathcal{F} = \{up(1), up(2), open\}$.
- actions: $\mathcal{A} = \{toggle(1), toggle(2)\}$.
- symbol: any $p \in \mathcal{F} \cup \mathcal{A}$.
  $range(p) =$ possible values (usually boolean $\{\mathtt{t}, \mathtt{f}\}$).

Action domains

- Value fact: $holds(p, v)$. Boolean abbreviations

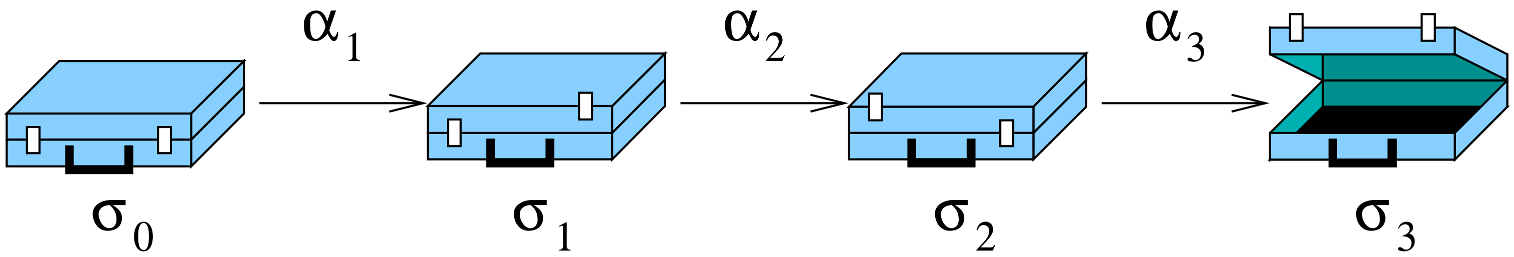
$$\begin{aligned} open &\stackrel{\text{def}}{=} holds(open, \mathtt{t}), \\ \overline{open} &\stackrel{\text{def}}{=} holds(open, \mathtt{f}) \end{aligned}$$

- State $\sigma : \mathcal{F} \longrightarrow \mathcal{V}$.
  Example: $\sigma_0 = \{\overline{up(1)}, up(2), \overline{open}\}$.

- Compound action $\alpha : \mathcal{A} \rightharpoonup \mathcal{V}$.
  Examples: $\alpha_1 = \{toggle(1)\}, \alpha_2 = \emptyset, \alpha_3 = \{toggle(1), toggle(2)\}$.

Narrative: $\sigma_0, \quad \alpha_1, \sigma_1, \quad \ldots, \quad \alpha_{n-1}, \sigma_{n-1}, \quad \alpha_n, \sigma_n$

$\{toggle(2)\}$ $\quad$ $\{toggle(1), toggle(2)\}$ $\quad$ $\{toggle(2)\}$



$\{\overline{up(1)}, \overline{up(2)}, \overline{open}\}$ $\quad$ $\{\overline{up(1)}, up(2), \overline{open}\}$ $\quad$ $\{up(1), \overline{up(2)}, \overline{open}\}$ $\quad$ $\{up(1), up(2), open\}$

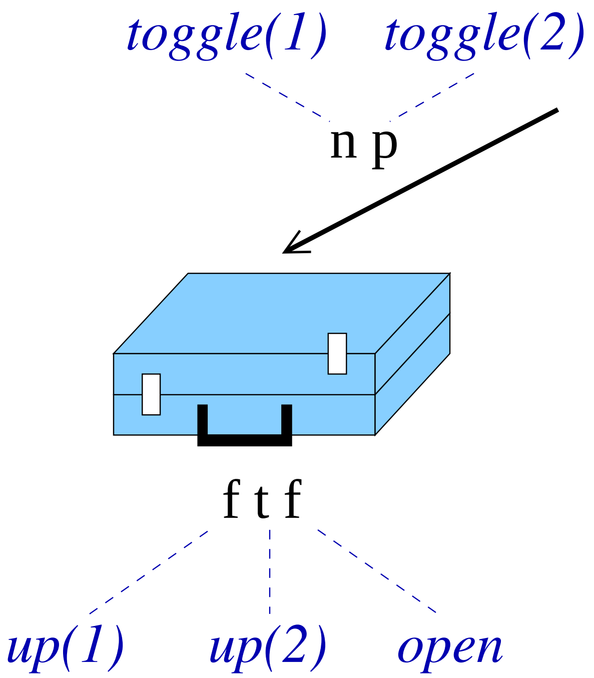
Atom: $holds(p, v, i)$, $i \in [0, n]$. Boolean abbreviations: p_i, $\overline{p_i}$.

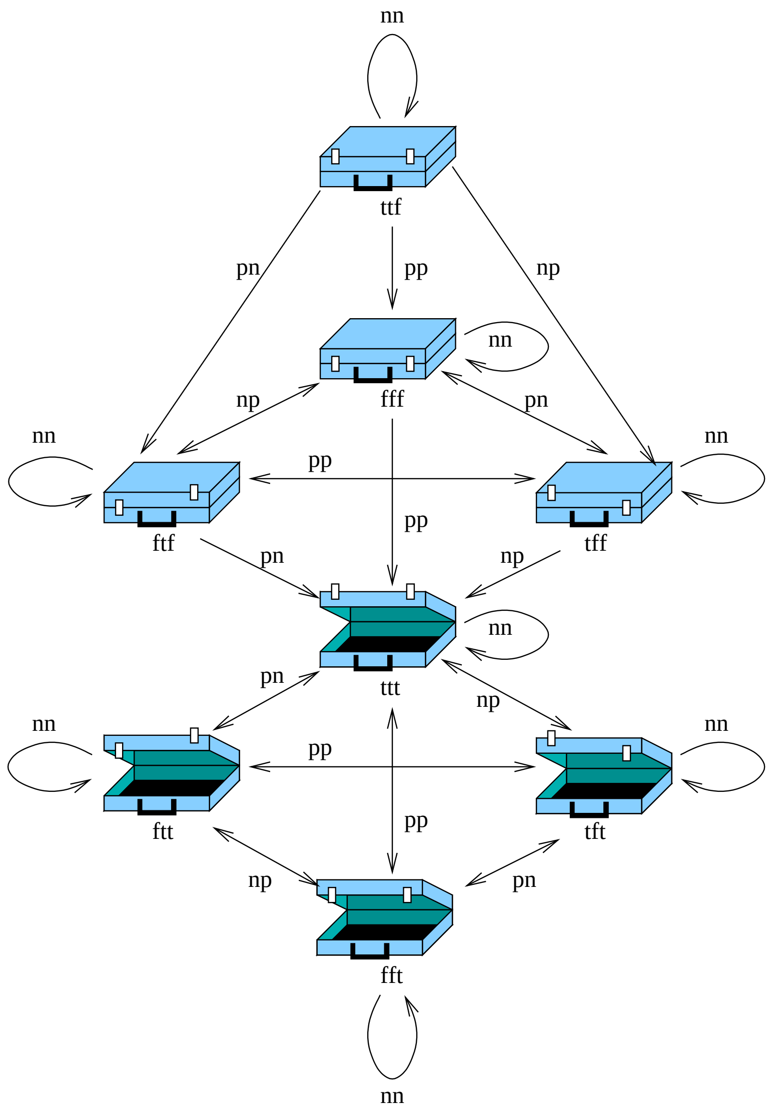
Transition systems

- σ_i only depends on σ_{i-1} and α_i.

- transition relation $\varrho \subseteq S \times A \times S$
  where $S =$ fluent states and $A =$ compound actions.

- We can use finite automata.


toggle(1)
toggle(2)
n p
f t f
up(1)
up(2)
open


nn
ttf
pn
pp
np
nn
fff
np
pn
nn
nn
pp
ftf
tff
pp
pn
np
nn
ttt
pn
np
nn
nn
pp
ftt
tft
pp
np
pn
fft
nn

Adding pertinence

- Pertinence facts $pert(p, v)$. $!p \stackrel{\text{def}}{=} pert(p, \mathtt{p})$ $\overline{!p} \stackrel{\text{def}}{=} pert(p, \mathtt{n})$

- Pertinence mapping $\pi : \mathcal{F} \cup \mathcal{A} \longrightarrow \{\mathtt{p}, \mathtt{n}\}$.

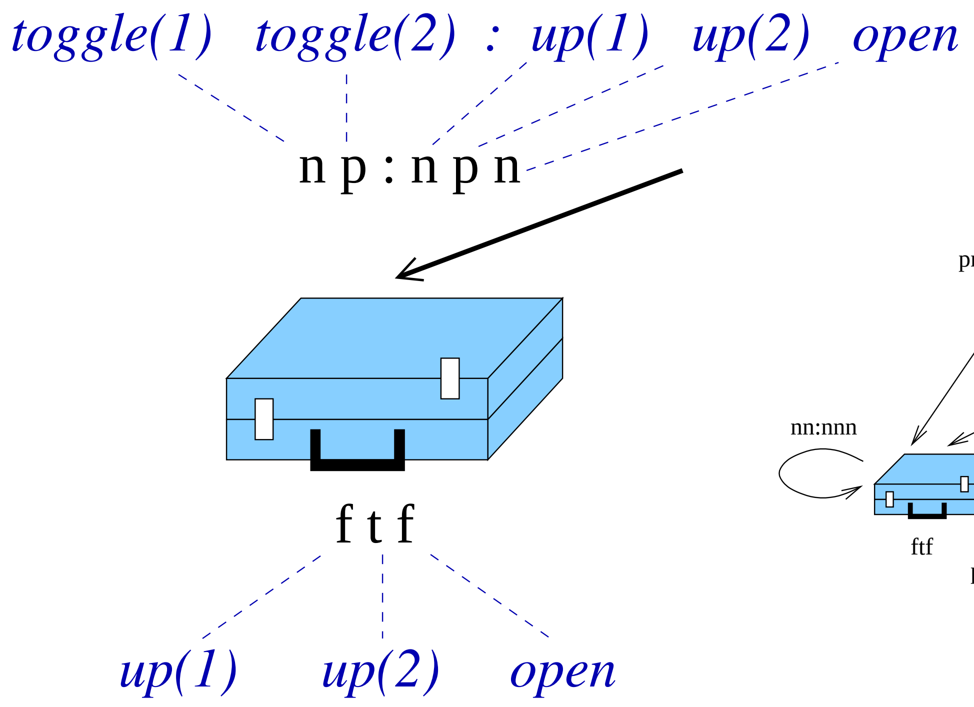
- Pertinence narrative: $\sigma_0, \quad \alpha_1, \sigma_1, \pi_1, \quad \ldots, \quad \alpha_n, \sigma_n, \pi_n$

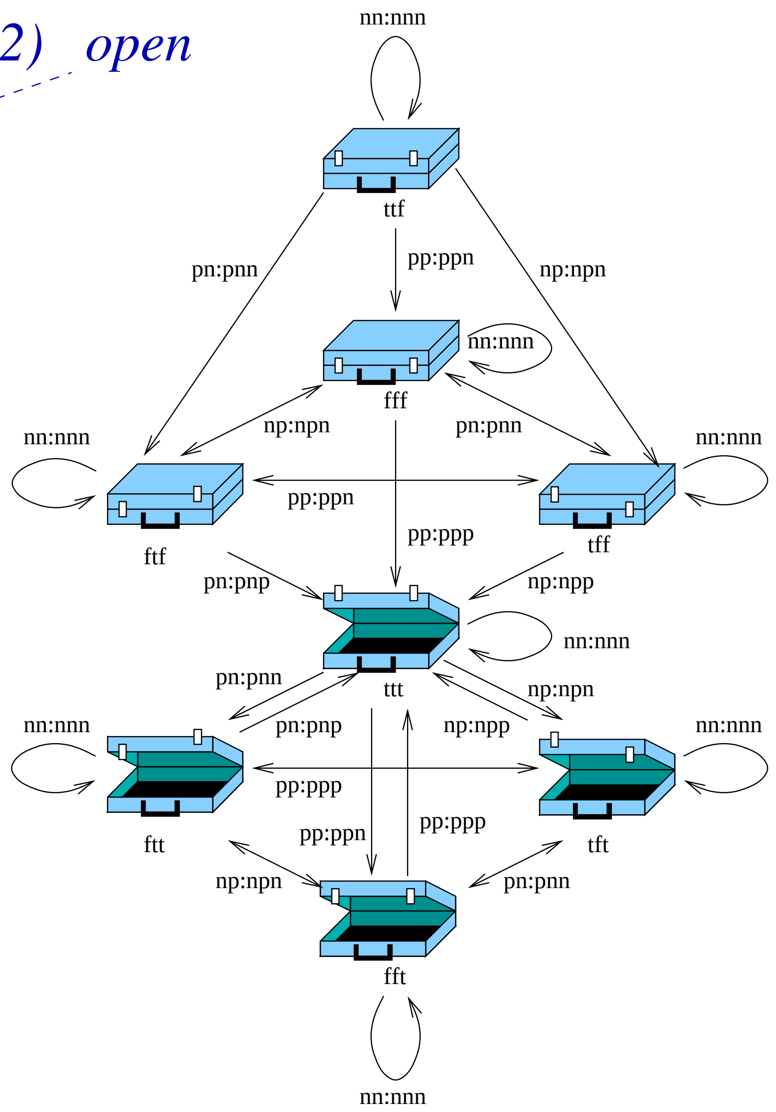
C1) an action is pertinent iff it is performed

C2) at 0, everything non-pertinent

C3) if $\pi_i(f) = \mathtt{n}$, then $\sigma_i(f) = \sigma_{i-1}(f)$.

- Transition systems. Pertinence as an output relation $\omega \subseteq S \times A \times P$ (where $P =$ pertinence mappings).


toggle(1) toggle(2) : up(1) up(2) open
n p : n p n
f t f
up(1) up(2) open


nn:nnn
ttf
pn:pnn
pp:ppn
np:npn
nn:nnn
fff
nn:nnn
np:npn
pn:pnn
nn:nnn
ftf
pp:ppn
pp:ppp
tff
pn:pnp
np:npp
nn:nnn
ttt
pn:pnn
np:npn
nn:nnn
pn:pnp
np:npp
nn:nnn
pp:ppp
ftt
pp:ppn
pp:ppp
tft
np:npn
pn:pnn
fft
nn:nnn

Automata

- ✔ full description of system behavior
- ✔ prediction, explanation and planning is straightforward
- ✘ very sensitive to representation changes
  (no elaboration tolerance)
- ✘ no way to detect causal interactions

Outline

- ✔ Introduction
- ✔ Pertinence Logic L^2
- ✔ Action domains and transition systems
- ☞ **$\mathcal{P}$-language: operational semantics**
- ❑ Logical formalization
- ❑ Causal cycles
- ❑ Related work
- ❑ System PAL
- ❑ Conclusions

$\mathcal{P}$-language

- Inspired by $\mathcal{A}$-language family [Gelfond&Lifschitz98]
- Rule shape: $\boxed{E \textbf{ if } C \textbf{ after } D}$

  E (effect): either $holds(f, v)$ or $\bot$,

  C (condition): $C_1 \wedge \cdots \wedge C_k$ (C_i facts of any kind),

  D (precondition): $D_1 \wedge \cdots \wedge D_m$ (D_i fluent value facts).

  We require no cycles among effects and conditions.

- Example (L=any lock):

$$
\begin{array}{cccccccc}
 & & & & up(L) & \textbf{if} & toggle(L) \textbf{ after } \overline{up(L)} & \\
 & & & & \overline{up(L)} & \textbf{if} & toggle(L) \textbf{ after } up(L) & \\
open & \textbf{if} & up(1) & \wedge & up(2) & & 0 & \leftarrow layer \\
2 & & 1 & & 1 & & &
\end{array}
$$

$\mathcal{P}$-language

- How to compute transitions? $\sigma_0, \alpha_1 \longrightarrow \sigma_1, \pi_1$
  increment a partial σ_1, π_1 applying rules layer by layer.

- A rule whose effect has layer j is applicable when:

  1. all the $D_i \in \sigma_0$, all $C_i \in \alpha_1$ (for actions)
  2. all the $C_i \in (\sigma_1 \cup \pi_1)$ up to $j - 1$
  3. for some symbol p in C: $!p \in \pi_1$ up to $j - 1$

Example

$\sigma_0 = \{\overline{up(1)}, up(2), \overline{open}\}$ $\alpha_1 = \{toggle(1)\}$

$$\begin{array}{lclclcll}
 & & & & up(L) & \textbf{if} & toggle(L) & \textbf{after}\ \overline{up(L)} \\
 & & & & \overline{up(L)} & \textbf{if} & toggle(L) & \textbf{after}\ up(L) \\
open & \textbf{if} & up(1) & \wedge & up(2) & & 0 & \leftarrow layer \\
2 & & 1 & & 1 & & &
\end{array}$$

layer j	Rule outcome	Persistence
0		$\emptyset$
1	$\{up(1)\}$	$\{up(2)\}$
2	$\{open\}$	$\emptyset$

procedure $transition(\sigma_0, \alpha_1,$**var** $\sigma_1,$**var** $\pi_1)$

begin

$Out := \emptyset;$

$Pers := \{holds(f, v) \in \sigma_0 \mid layer(f) = 0\};$

$\sigma_1 := Pers;$

$\pi_1 := \{!a \mid occurs(a, \alpha_1)\} \cup \{\overline{!a} \mid \text{not } occurs(a, \alpha_1)\} \cup$
$\{\overline{!(f)} \mid layer(f) = 0\};$

for j:=0 **to** $max(layer) - 1$ **do**

$Out := \{E \mid E$ effect of some applicable rule at layer $j\};$

if $inconsistent(Out)$ **or** $\perp \in Out$ **then**

halt("α_1 not executable");

$Pers := \{holds(f, v) \in \sigma_0 \mid layer(f) = j, \text{not } occurs(f, Out)\};$

$\sigma_1 := \sigma_1 \cup Out \cup Pers;$

$\pi_1 := \pi_1 \cup \{!f \mid occurs(f, Out)\} \cup \{\overline{!f} \mid occurs(f, Pers)\}$

end-for

end

Outline

- ✔ Introduction
- ✔ Pertinence Logic L^2
- ✔ Action domains and transition systems
- ✔ $\mathcal{P}$-language: operational semantics
- ☞ **Logical formalization**
- ❑ Causal cycles
- ❑ Related work
- ❑ System PAL
- ❑ Conclusions

Pertinence Calculus: axioms AX

- Two predicates: $holds(F,V,I) \quad pert(F,V,I)$.
- Domain closure and unique names assumption for $\mathcal{A}, \mathcal{F}, \mathcal{V}, [0,n]$.
- Uniqueness and existence of value:

$$\forall P, I, V, V'.\ \Big(V \neq V' \supset \neg(holds(P,V,I) \wedge holds(P,V',I))\Big)$$

$$\forall F, I.\ \exists V.\ holds(F,V,I)$$

$$\forall P, I.\ \Big(pert(P,\mathtt{n},I) \equiv \neg pert(P,\mathtt{p},I)\Big)$$

- Conditions C1, C2, and C3 on narratives:

$$\forall A, I.\ \Big(pert(A,\mathtt{p},I) \equiv \exists V.\ holds(A,V,I)\Big)$$

$$\forall P.\ pert(P,\mathtt{n},0)$$

$$\forall F, V, I.\ \Big(\ I > 0\ \wedge pert(F,\mathtt{n},I) \wedge holds(F,V,I-1) \supset holds(F,V,I)\Big)$$

(UFR)

Pertinence calculus: rule translations $t(R)$

$\boxed{E \ \textbf{if}\ C \ \textbf{after}\ D}$ becomes $\boxed{\forall I.\big(I > 0 \land D_{I-1} \land \mathbb{C}\, C_I \supset \mathbb{C}\, E_I\big)}$

(L^2 operators $\mathbb{C}$, !, etc are "unfolded.")

$$\underbrace{up(1)}_{E}\ \textbf{if}\ \underbrace{toggle(1)}_{C}\ \textbf{after}\ \underbrace{\overline{up(1)}}_{D}$$

$$\Downarrow$$

$$\forall I.\big(I > 0 \land \underbrace{\overline{up(1)}_{I-1}}_{D_{I-1}} \land \underbrace{toggle(1)_I \land !toggle(1)_I}_{\mathbb{C}\, C_I} \supset \underbrace{up(1)_I \land !up(1)_I}_{\mathbb{C}\, E_I}\big)$$

$$\underbrace{open}_{E}\ \textbf{if}\ \underbrace{up(1) \land up(2)}_{C}$$

$$\Downarrow$$

$$\forall I.\big(I > 0 \land \underbrace{up(1)_I \land up(2)_I}_{C_I} \land \underbrace{(!up(1)_I \lor !up(2)_I)}_{!C_I} \supset \underbrace{open_I \land !open_I}_{\mathbb{C}\, E_I}\big)$$

Other constructions

For prediction problems,
we define complete action observations **do** (α, i).

$$\textbf{do}\ (\{toggle(2)\}, 4) \stackrel{\text{def}}{=} toggle(2)_4 \wedge \neg !toggle(1)_4$$

$$\textbf{do}\ (\emptyset, 5) \stackrel{\text{def}}{=} \neg !toggle(1)_5 \wedge \neg !toggle(2)_5$$

axioms+rules are not enough!

- Example: $\overline{up(1)}_0 \wedge up(2)_0 \wedge \overline{open}_0 \wedge$ **do** $(\{toggle(1)\}, 1)$

- Our logical theory yields 5 models (*transition* yields M_4)

	$up(2)_1$	$open_1$
M_1	`fp`	`fn`
M_2	`fp`	`fp`
M_3	`fp`	`tp`

	$up(2)_1$	$open_1$
M_4	`tn`	`tp`
M_5	`tp`	`tp`

- We need formulas for what does not change (frame axioms). Problems of elaboration tolerance: unrelated facts, concurrent actions, indirect effects.

- Result: obtaining frame axioms by "inspection" is unfeasible!

Solution: inertia default

- *"By default, fluents must remain unchanged."*
  ... but this means nonmonotonicity. Example:
  - When $dark\_room_0$, open the suitcase: $dark\_room_1 \in Cn(T)$.
  - Now, there is an internal lamp inside the suitcase:

$$dark\_room_1 \notin Cn(T \cup \{\overline{dark\_room} \textbf{ if } open\}) \tag{2}$$

- Techniques:
  - Circumscription
  - Default logic
  - Logic Programming: Stable Models, Completion, Well-founded semantics.

Circumscription [McCarthy80]

- Minimize the extent of a predicate ($pert$).

- Syntactically, a second order formula:

$$\mathrm{CIRC}[T; pert] \stackrel{\text{def}}{=} T(pert) \wedge \neg\exists P.\ (T(P) \wedge P < pert)$$

- Semantically, minimal models:

  take the less extent of $pert$ from models with fixed $holds$.

- We use the policy:
  $\mathrm{CIRC}[t(R) \cup Obs \cup \mathrm{AX}';\ pert] \quad \cup \quad (\mathrm{UFR})$

Circumscription

- Example:

$$\overline{up(1)}_0 \wedge up(2)_0 \wedge \overline{open}_0 \wedge \textbf{ do } (\{toggle(1)\}, 1)$$

- Models selection:

	$up(2)_1$	$open_1$	
M_6	fn	fn	minimal
M_7	fn	fp	
M_1	fp	fn	
M_2	fp	fp	
M_8	fn	tn	minimal
M_9	fn	tp	
M_{10}	fp	tn	
M_3	fp	tp	
M_4	tn	tp	minimal
M_5	tp	tp	

Circumscription

Theorem 7
Circumscription policy corresponds to the operational semantics, provided that rules do not contain $\overline{!p}$ in their conditions.

- Example: the alarm scenario.

$$\begin{array}{lll} active & \textbf{if} & connect \\ \overline{active} & \textbf{if} & disconnect \end{array} \qquad \begin{array}{rll} in & \textbf{if} & enter \\ ring & \textbf{if} & in \wedge active \wedge \overline{!active} \end{array}$$

- Executing action $\{enter\}$ we get a model with $M(active) = \mathtt{tp}$.

Default Logic [Reiter80]

- A default is a special inference rule of shape:

$$\frac{\alpha : \beta_1, \ldots, \beta_m}{\gamma}$$

- Reduct D^E (D set of defaults, E logically closed theory):

$$D^E = \left\{ \frac{\alpha}{\gamma} \;\middle|\; \frac{\alpha : \beta_1, \ldots, \beta_m}{\gamma} \in D, \text{ and there is no } \neg\beta_j \in E \right\}$$

- D^E only contains "usual" inference rules.
  $Cn(D^E)$ denotes its deductive closure.

- E is an extension of D iff $E = Cn(D^E)$.

Pertinence in Default Logic

Rule translation

$$\frac{D_{I-1} \wedge \mathbb{C}\, C_I}{\mathbb{C}\, E_I}$$

(UFR)

$$\frac{pert(F, \mathtt{n}, I) \wedge holds(F, V, I-1)}{holds(F, V, I)}$$

Pertinence minimization

$$\frac{: pert(P, \mathtt{n}, I)}{pert(P, \mathtt{n}, I)}$$

- Default Logic fully captures the operational behavior, but for practical purposes, we can use its strong connection to stable models.

Logic Programming (LP).

A pertinence calculus theory will become a logic program to be interpreted under:

1. Stable models

2. Clark's completion

3. Well-founded semantics

Stable models [Gelfond&Lifschitz88]

- Like default theories, but restricted to atoms:

  $\dfrac{a_1 \wedge a_2 : \neg b_1, \neg b_2}{c}$ becomes the rule $c \leftarrow a_1, a_2, \mathit{not}\ b_1, \mathit{not}\ b_2$

- Reduct P^M (P set of rules, M set of ground atoms):

  $$P^M = \{(c \leftarrow a_1, \ldots, a_n) \mid (c \leftarrow a_1, \ldots, a_n, \mathit{not}\ b_1, \ldots, \mathit{not}\ b_m) \in P, \text{ and all } b_j \notin M\}$$

- P^M has not default negation. It has a least model (closure of direct consequences) written $\Gamma_P(M)$.

- Stable models are fixpoints: $M = \Gamma_P(M)$.

Stable models

Some properties:

- We may get several stable models:

$$\begin{array}{rcl} p & \leftarrow & \textit{not}\ q \\ q & \leftarrow & \textit{not}\ p \end{array}$$

- ... or no stable model at all:

$$p \leftarrow \textit{not}\ p$$

- ... but only in presence of "negative" cycles.

Well-founded semantics (WFS) [van Gelder et al. 91]

- Γ_P^2 has a least and a greatest fixpoint.
  (both computable by iteration).

- Well-founded model (WFM):
  - True atoms: $lfp(\Gamma_P^2)$
  - False atoms: $\Sigma - gfp(\Gamma_P^2)$
  - the rest are left undefined (due to negative cycles)

- defined atoms in the WFM are fixed in all the stable models.

- For explicit negation $\overline{p}$, a variant is needed:
  WFSX [Alferes&Pereira92].

Clark's completion [Clark78]

- Consider the program P_1:

$$\begin{array}{lll} p & \leftarrow & q, r \\ p & \leftarrow & s \\ r & & \end{array} \qquad\qquad \begin{array}{lll} P_1 & \not\models & p \\ P_1 & \not\models & \neg p \end{array}$$

- Idea: rules are the only ways to obtain head atoms.
  $\mathrm{COMP}[P_1] = \{p \equiv (q \wedge r) \vee s, \quad r \equiv \top, \quad q \equiv \bot, \quad s \equiv \bot\}$

- Its models (called supported) are the fixpoints of

$$T_P(M) = \{c \mid (c \leftarrow Body) \in P \text{ and } M \models Body\}$$

- Positive cycles: $\{p \leftarrow p\}$ has 2 supported models $\{p\}$ and $\emptyset$.

Logic Programming

- A program is hierarchical iff it contains no cycles.

- Interesting property:

  For a hierarchical programs, stable models, Clark's completion and WFS yield the same unique model.

Pertinence in Logic Programming

- Axioms:

$$
\begin{array}{rcl}
\bot & \leftarrow & holds(P,V,I),\ holds(P,V',I) \\
\bot & \leftarrow & pert(P,\mathtt{p},I),\ pert(P,\mathtt{n},I) \\
pert(A,\mathtt{p},I) & \leftarrow & holds(A,V,I) \\
holds(F,V,I) & \leftarrow & pert(F,\mathtt{n},I),\ holds(F,V,I-1)
\end{array} \qquad \text{(UFR)}
$$

- Pertinence minimization: $pert(P,\mathtt{n},I) \leftarrow$ *not* $pert(P,\mathtt{p},I)$

- Example of $\mathcal{P}$-rule translation: $open$ **if** $up(1) \wedge up(2)$ becomes

$$
\begin{array}{rcl}
open_I & \leftarrow & up(1)_I,\ up(2)_I,\ !up(1)_I \\
!open_I & \leftarrow & up(1)_I,\ up(2)_I,\ !up(1)_I \\
open_I & \leftarrow & up(1)_I,\ up(2)_I,\ !up(2)_I \\
!open_I & \leftarrow & up(1)_I,\ up(2)_I,\ !up(2)_I
\end{array}
$$

Pertinence in Logic Programming

Theorem 8
Given a set of acyclic $\mathcal{P}$-rules R, the corresponding logic program is hierarchical.

Proof sketch: the resulting program $P(R)$ can be arranged into different levels (3 levels per layer).

Theorem 9
The logic programming encoding under (any of the 3 LP semantics) corresponds to the operational semantics.

Proof sketch: proceed by induction on situations and layers, using WFS iterative construction.

Then, why 3 semantics? $\Rightarrow$ they differ for cycles.

Outline

- ✔ Introduction
- ✔ Pertinence Logic L^2
- ✔ Action domains and transition systems
- ✔ $\mathcal{P}$-language: operational semantics
- ✔ Logical formalization
- ☞ **Causal cycles**
- ❑ Related work
- ❑ System PAL
- ❑ Conclusions

Causal cycles

- Cyclic $\mathcal{P}$-rules $\Rightarrow$ layer-based algorithm not applicable.

- Depending on the non-monotonic technique, we may obtain: non-existence of model, non-determinism, undefined values.

- Most effects will depend on the studied transition (cycles behave dynamically).

Self-supportedness

- Problem: the ultimate justification for an effect should not be the effect itself.

- Example: having $\overline{b}_0 \wedge$ **do** $(\emptyset, 1)$ and rule: $\boxed{b \textbf{ if } b}$

  1. Circumscription, WFS, stable models: b persists false.

  2. Completion yields 2 models (non-determinism):
     (the one above + b becomes caused true!).

- b is "self-justified" without external intervention.

Self-supportedness

- Also in circumscription ...

- Example: $\boxed{b \textbf{ if } b \wedge a}$ with $\bar{b}_0 \wedge$ **do** $(\{a\}, 1)$.

  1. WFS, stable models: b persists false.

  2. Completion, Circumscription: additional model with b caused true.

- Important property: stable models guarantee absence of self-supportedness (and in fact, the same applies for WFS).

Non-existence of model

- Example: $\boxed{b \textbf{ if } b \wedge a}$ with $b_0 \wedge$ **do** $(\{a\}, 1)$.

  1. stable models: no stable model.

  2. WFS: atoms b_1, $!b_1$ and $\overline{!b_1}$ become undefined.

  3. Circumscription, completion: b caused true (self-supported).

- Using these kind of cycles for non-executability of actions is not a good idea:

  - their behavior depends on the previous situation,

  - we can use: $\perp$ **if** a **after** b

  Better to use this as hint of bad design.

WFS always “tries” determinism

$$\boxed{\begin{array}{lll} b & \textbf{if} & \overline{c} \wedge a \\ c & \textbf{if} & \overline{b} \wedge a \end{array}}$$

$$\overline{b}_0 \wedge \overline{c}_0 \wedge \ \textbf{do}\ (\{a\}, 1)$$

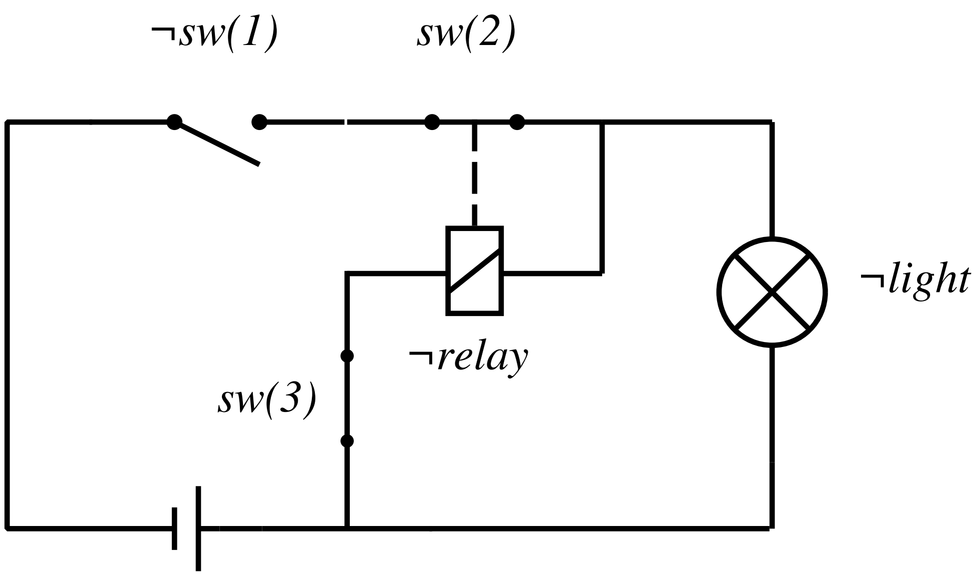
- Result:

  1. WFS: b and c undefined.
  2. the rest yield 2 solutions:
     b persists false, c caused true; and vice versa.

- For other initial states, all semantics agree.

- Again: unclear way for intended non-determinism.
  It seems to point out bad design.

Example of “realistic” cycle problem



Relay circuit [Shanahan99]

- Toggling switch 1 at the situation above:
    1. WFS: $sw(2), relay$ and $light$ undefined ($sw(1)$ caused true).
    2. the rest: no model.

Outline

- ✔ Introduction
- ✔ Pertinence Logic L^2
- ✔ Action domains and transition systems
- ✔ $\mathcal{P}$-language: operational semantics
- ✔ Logical formalization
- ✔ Causal cycles
- ☞ **Related work**
- ❑ System PAL
- ❑ Conclusions

Related action approaches

1. Change as abnormality
2. Lin's $caused$ predicate
3. Occlusion/TAL
4. Event Calculus
5. Inductive Causation
6. Abnormality in Logic Programming
7. Thielscher's Causal Relationships
8. Causal Explanation
9. Schwind's comparative

2. Lin's *caused* predicate [Lin95]

- Truth value reified:

$$\begin{aligned} caused(F, \mathtt{t}, S) &\supset holds(F, S) \\ caused(F, \mathtt{f}, S) &\supset \neg holds(F, S) \end{aligned}$$

- $caused$ versus persistent:

$$\neg caused(F, \mathtt{t}, S) \wedge \neg caused(F, \mathtt{f}, S) \supset [holds(F, do(A, S)) \equiv holds(F, S)] \quad \text{(UFR)}$$

- Similar circumscription policy (leave (UFR) outside).
- no P5: conditions do not refer to $caused$

$$holds(up(1), S) \wedge holds(up(2), S) \supset caused(open, \mathtt{t}, S)$$

- no P1: waiting, when the suitcase is already open.
- unlike $pert$, $caused$ is not defined for formulas.

3. Occlusion [Sandewall94]/TAL [Gustafsson&Doherty96]

- In [Sandewall94] (page 235), we find P4:
  "...occlusion does not *imply* change; it merely allows it."

$$\neg occlude(F, I) \supset holds(F, I-1) \equiv holds(F, I) \qquad \text{(UFR)}$$

- Same circumscription (minimize $occlude$ with (UFR) outside).

- In low-level notation, the rule for $open$ becomes:

$$\begin{aligned} \overbrace{holds(up(1), I) \wedge holds(up(2), I)}^{C_I} \quad &\supset \quad holds(open, I) \\ C_I \wedge \neg C_{I-1} \quad &\supset \quad occlude(open, I) \end{aligned}$$

- P5 not satisfied, but change of truth value in the condition.
  Occlusion of ϕ: all fluents in ϕ occluded ($\neq!\phi$).

5. Inductive Causation [Denecker et al.98].

- P1 is satisfied ([Denecker et al.98] section 5.3, page 24):

  "...an effect is never triggered by the absence of other effects alone, e.g. the suitcase will not be open just because nothing happens."

- Alternative underlying non-monotonic techniques (cycles).

- $\boxed{\textbf{initiating } up(1) \wedge up(2) \textbf{ causes } open}$ becomes:

$$
\begin{aligned}
caus(open) &\leftarrow init(up(1)) \wedge holds(up(2)) \wedge \neg init(\neg up(2)) \\
caus(open) &\leftarrow init(up(2)) \wedge holds(up(1)) \wedge \neg init(\neg up(1)) \\
caus(open) &\leftarrow init(up(1)) \wedge init(up(2))
\end{aligned}
$$

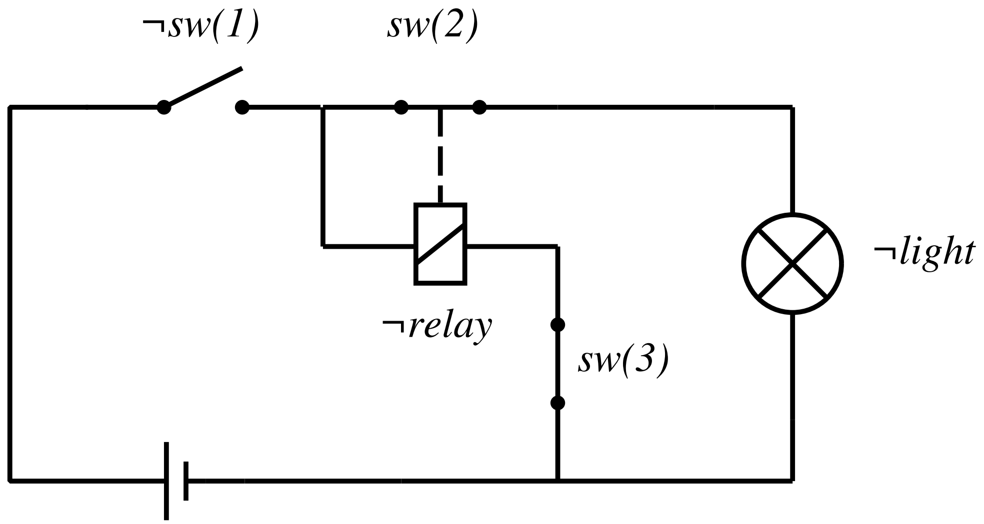
- Initiation of a formula $\Rightarrow$ close to pertinence!

- However, no P5: *init* means an actual change of value.

6. Causal Relationships [Thielscher97].

$$\boxed{\begin{array}{l} up(1) \textbf{ causes } open \textbf{ if } up(2) \\ up(2) \textbf{ causes } open \textbf{ if } up(1) \end{array}}$$

- Algorithmic: trigger rules with intermediate states.
- During the process, fluent may change its value.
- It can be non-deterministic even with acyclic rules.



7. Causal Explanation [McCain&Turner97].

- Strongly related to completion/supported models.

- $\phi \leftarrow \psi \quad \approx \quad \psi$ true implies ϕ "caused."

- Strictly inferential (no change predicate).

$$\begin{array}{rcl} holds(open, I) & \leftarrow & holds(up(1), I) \wedge holds(up(2), I) \\ holds(F, I+1) & \leftarrow & holds(F, I+1) \wedge holds(F, I) \\ \neg holds(F, I+1) & \leftarrow & \neg holds(F, I+1) \wedge \neg holds(F, I) \end{array}$$

- Important: "*caused*" has a different meaning here.
  Everything must get finally "caused," including inertia!

Outline

- ✔ Introduction
- ✔ Pertinence Logic L^2
- ✔ Action domains and transition systems
- ✔ $\mathcal{P}$-language: operational semantics
- ✔ Logical formalization
- ✔ Causal cycles
- ✔ Related work
- ☞ **System PAL**
- ❑ Conclusions

PAL

- Extended language: exploit functional notation

```
up(L):=not prev(up(L)) if toggle(L);
```

- Main modules (programmed in `C`):

1. Rule/query grounding:
   - ground $\mathcal{P}$-rules
   - logic programs in SMODELS internal format (`C++`)
2. Action generator (for temporal queries)
3. WFSX algorithm

Outline

- ✔ Introduction
- ✔ Pertinence Logic L^2
- ✔ Action domains and transition systems
- ✔ $\mathcal{P}$-language: operational semantics
- ✔ Logical formalization
- ✔ Causal cycles
- ✔ Related work
- ✔ System PAL
- ☞ **Conclusions**

Conclusions

- Methodology: begin with automata, $\mathcal{P}$-language and operational semantics.
- Logical formalization at 2 levels:
  1. Theoretical foundation: pertinence, L^2.
  2. Several alternatives for nonmonotonic inference (circumscription, stable models, completion, WFS).
- Comparisons (cycles, other action aproaches).
- PAL interpreter.

Future work

1) Usual topics in Reasoning about Actions:
(Explicit) nondeterminism, qualification problem, natural actions, actions with durations, sensing and robot knowledge.

2) Other interesting topics:

- Strong equivalence between causal representations. [Lifschitz,Pearce,Valverde00]
- Proving temporal properties.
- Machine Learning [Lorenzo01].

3) PAL: improvements + extensions.

Resumen