On Modelling of Inertia in Action Languages.

Mikhail Prokopenko

CSIRO
Division of Information Technology,
Locked Bag 17, North Ryde, NSW 2113,
Australia
E-mail: mikhail@syd.dit.csiro.au

Pavlos Peppas

Department of Computing,
School of MPCE,
Macquarie University, NSW 2109,
Australia
E-mail: pavlos@ mpce.mq.edu.au

1. Introduction.

Various investigations of the ramification problem [4, 6 and others] have demonstrated that in order to deduce the indirect effects of actions it is eminent to *efficiently* represent a domain's background knowledge. The notion of efficiency is usually based on the idea of compact and concise representation and dates back at least as far as Occam's Razor: other things being equal, simple theories are to be preferred to complex ones. For instance, in the framework of first-order logic and its nonmonotonic extensions, representation of background knowledge in the form of domain constraints [6] is preferable in respect to an explicit consequences approach (such as STRIPS [3]) because it avoids exponential difficulties usually associated with the latter[1]. Appropriately represented knowledge at the same time has to be *efficiently* "readable" and "processable" by an underlying reasoning mechanism. Such a reasoning system[2] should be capable of inferring both direct and indirect consequences of a performed action (an occurred event) as well as preserve inertial properties of a domain (solve the frame problem [16]).
Action theories recently developed in the framework of action languages with inertia and ramifications [11, 7, 10, 8] use the idea of minimising change to deduce the set of possible next states (successor states). The notion of minimal change is usually defined by set inclusion and incorporates the concept of frame, assigning different degrees of inertia for language elements (fluents, literals, formulas, etc.). For example, in [7] it is noted that "if F is not a frame fluent then it is not expected to keep its old value after performing an action, so that the change in its value is disregarded". In addition, some action theories try to represent domain knowledge in a way that not only provides a solution to the frame and the ramification problems but also embodies background information in the form of "causal laws" of a domain. Moreover, it was successfully argued in many publications [2, 12, 15, 19] that, in general, propositions embracing causal dependencies are "more expressive than traditional state constraints" [19]. Following recent approaches to representing causal information, we do not intend here to investigate philosophical aspects of the phenomenon - probably discovering, understanding, and formalising an elusive nature of the cause-effect relation will by itself mark a significant success on the way to engineer artificial intelligence. However, we shall try to highlight possible areas of interaction between such ontological characteristics of action domains as inertia and causality and to answer the following questions in particular:

[1] It is not trivial to find a set of domain constraints always producing desirable inferences in the domain, and complexity of this task sometimes is comparable with that of maintaining satisfactory descriptions of the actions in an explicit consequences approach [18].
[2] The term "reasoning system" is used to refer to "any formal system that produces inferences about the effects of events" [17].

- under what conditions do we gain by dividing fluents on inertial and non-inertial ones in the framework of an action language with fluent-triggered causality[3] (fact causality) and
- how does it affect possible solutions to the ramification problem.

We believe that the best way to proceed towards this goal is to trace the evolution of action languages.

2. Evolution of Action Languages.

As mentioned in [14], the original idea behind introducing action languages was to present a methodology allowing for translation from a specialised action language to a general-purpose formalism, such as a nonmonotonic reasoning system based on first-order logic. A domain described in the first action language *A* [5], for instance, can be translated [9] into a logic programming language or into the circumscriptive approach of Baker [1]. Analogously [11] defines a translation from AR_0 , another action language, into the formalism of nested abnormalities theories [13]. Subsequent research, however, has shown that in addition, an advantage of having strict syntax and clear semantics transforms action languages into useful tools for better understanding of the aspects of reasoning about actions. Hence "defining action languages, comparing them and studying their properties" [14] can help in "capturing our commonsense intuitions about the whole family of action domains expressible in the language" [19].

2.1 AR Languages.

Consider the oldest and better developed branch of the action languages' "evolutionary tree" that can be represented as follows:

$$A \dashrightarrow AR^{-} \dashrightarrow AR_0 \dashrightarrow AR \dashrightarrow ARD.$$

The possibility of ramifications is the main feature [11] of all "*AR*"-dialects of *A*. Despite the fact that the AR^{-} language [10] appeared chronologically later than the AR_0 [11] we treat the former as evolutionary predecessor of the latter. Syntactically, AR^{-} is a subset of AR_0, but most importantly, the classes of domains expressible in the languages relate to each other likewise. The notions of expanding expressibilitiy of action languages and classes of domains is central in ordering of languages and constructing the tree. All mentioned languages are capable of describing domains with deterministic actions, truth-valued fluents, inertia, and action-triggered causality. In addition, they evolve by expanding the domains properties as shown in the following table:

	A	AR^{-}	AR_0	*AR*	*ARD*
non-deterministic actions			*	*	*
non-truth-valued fluents				*	*
inertial fluents		*	*	*	*
dependent fluents[4]					*
state constraints		*	*	*	*

[3] The terms action- and fluent-triggered causality are due to Lin [12].

[4] the fluents evaluation of which depends on their histories and values of other fluents [8].

One can expect that general-purpose formalisms to which action languages are translated to gain in complexity as well. A detailed investigation of such a parallel evolution, however, lies beyond the scope of this paper.
On the contrary, relationships among classes of domains covered by evolving action languages are of major interest. The expanding expressibility of the languages does not guarantee, by itself, enlargement of corresponding classes of domains. Nevertheless, in the analysed evolutionary branch one can observe that such a process is at least non-diminishing:

- AR^{-} adds inertial (frame) fluents and domains constraints to *A*; ie., "*A* domains can be thought of as a special case of deterministic AR^{-} domains where there are no constraints" [10];
- AR_0 broadens the classes of deterministic and temporal projection action domains expressible in AR^{-} by allowing simple forms of non-determinism;
- *AR* [7] introduces new language elements (non-propositional fluents) but "preserves the meaning" [7] of a domain description thus guaranteeing that classes of action domain did not reduce[5];
- the contribution of *ARD* [8] is not only in showing how to represent "non-persistent ramifications" but also in introducing "a history" in the language semantics - opening a way to formalise some of the action domains with "hypothetical reasoning".

2.1 AC Languages.

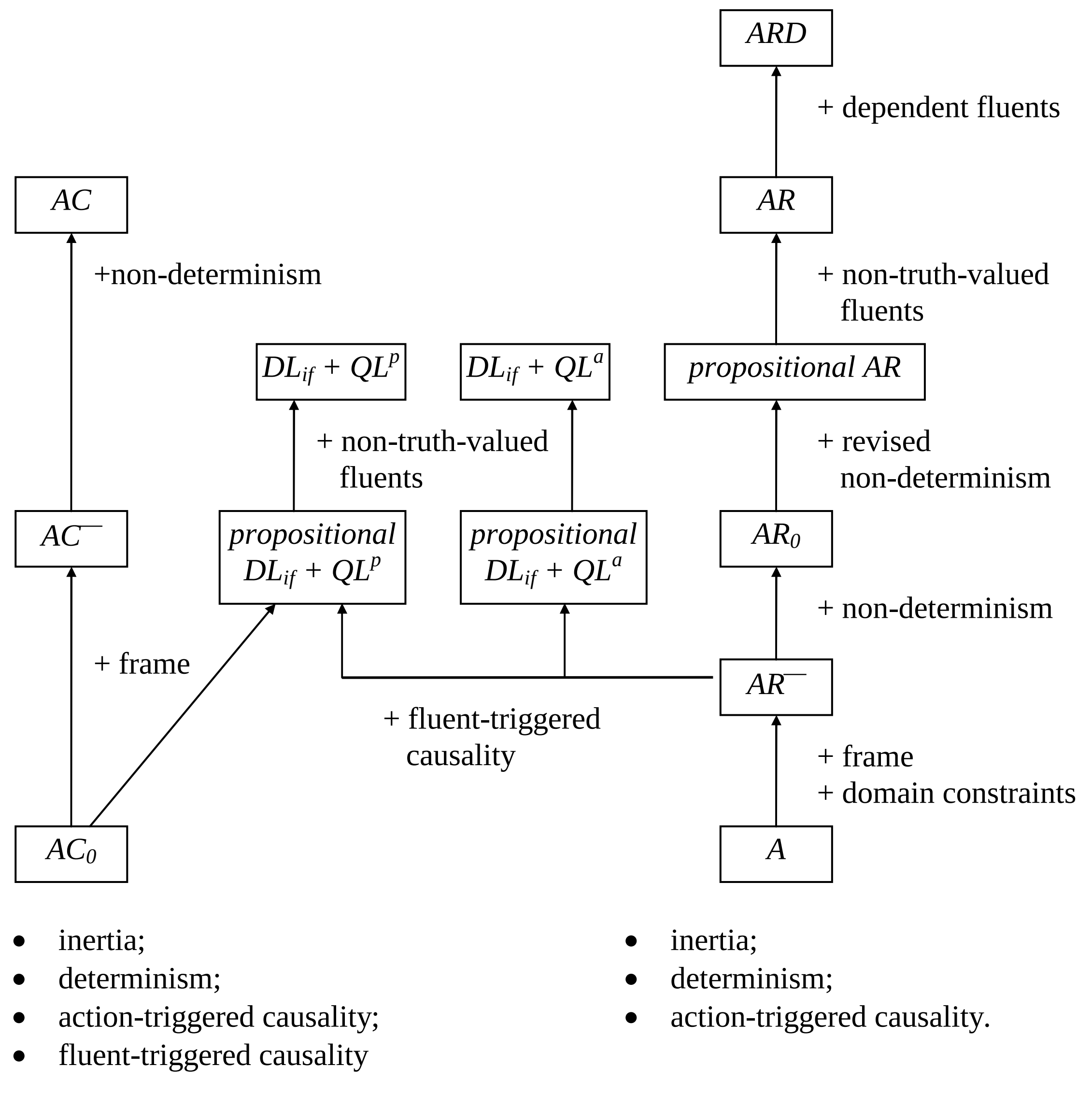
The *AR*-languages are characterised by the use of state constraints to represent background knowledge. Analogously, all *AC*-languages (called so because of the language *AC* introduced by Turner [19]) feature fluent-triggered (fact) causality in this role distinguishing themselves from the *AR*-languages.
The *AC* follows the approach for representing causal background knowledge that was proposed by McCain and Turner in [15] and was not formalised as an action language. Nevertheless, it is straight-forward to describe an action language, called let us say, AC_0 , based on the approach constructed in [15] that can serve as an origin of the *AC*-languages branch[6].
As an action language *AC* is modelled after a propositional version of the language *AR* and hence incorporates the idea of inertial fluents and the notion of non-determinism. In this paper we propose to augment the basic language AC_0 incrementally: first - by adding the concept of a frame, and second - by extending it to non-deterministic action domains. In other words, we propose to consider yet another subset of *AC*: the language AC^{-} that takes the intermediate place between the AC_0 and the *AC*. This suggestion has dual purpose - to present the evolution in a more steady way and to answer the questions regarding the role of the frame concept.
The evolution along the *AC*-line can be represented then in the following table:

	AC_0	AC^{-}	*AC*
non-deterministic actions			*
inertial fluents		*	*
fluent-triggered causality[7]	*	*	*

The action languages' evolutionary tree is shown in Figure 1. Since the propositional version of *AR* differs from AR_0 (different semantics of "release" / "possibly changes" propositions), the

[5] Influence propositions of *AR* are supposed to correct AR_0 's handling of non-deterministic actions.
[6] See the next section for the formal description of the AC_0 .
[7] state constraints are subsumed by fluent-triggered causality in both their roles: as restrictions on the set of states and as producers of indirect effects of actions.

“propositional *AR*” has been identified as a missing link of evolution that improves handling of non-deterministic actions compared to the AR_0 .
In addition, the tree includes a family of languages with fact causality introduced by Lifschitz in [14]. The languages describe deterministic action domains: the “$DL_{if} + QL^p$ “ language is aimed at temporal projection problems and the “$DL_{if} + QL^a$ “ language - at temporal explanation problems. Although these two languages allow to represent non-propositional fluents from the moment of their appearance it is quite natural to assume that evolutionary tree might include their propositional counterparts as well.



ARD
+ dependent fluents
AC
AR
+non-determinism
+ non-truth-valued fluents
DL_if + QL^p
DL_if + QL^a
propositional AR
+ non-truth-valued fluents
+ revised non-determinism
AC⁻
propositional DL_if + QL^p
propositional DL_if + QL^a
AR_0
+ non-determinism
+ frame
AR⁻
+ fluent-triggered causality
+ frame
+ domain constraints
AC_0
A
• inertia;
• determinism;
• action-triggered causality;
• fluent-triggered causality
• inertia;
• determinism;
• action-triggered causality.

Figure 1. Evolutionary tree of action languages.

3. A Comparison between AC_0 and AC^{-}.

3.1 AC_0 Language.

Following [15, 19], we can define an AC_0 language by

- a non-empty set of symbols Φ, that are called fluent names, or fluents and
- a non-empty set of symbols Ω, that are called action names, or actions.

In other words, an AC_0 language is set by its signature $< \Phi, \Omega >$.
A formula is a propositional combination of fluents. A fluent literal is an expression F or ¬F, where F is a fluent name. An AC_0 domain description consists of

- value propositions

φ **after** $\bar{A}$,

where φ is a fluent formula and $\bar{A}$ is a string of actions. If $\bar{A}$ is the empty string, value proposition is abbreviated as

initially φ;

- sufficiency propositions

φ **suffices for** ψ[8],

where φ and ψ are fluent formulas;

always φ

is an abbreviation for

True **suffices for** ψ,

- effect propositions

A **causes** φ **if** ψ,

where A is an action, and φ and ψ are fluent formulas.

Detailed description of the semantics of the AC_0 can be easily obtained from the corresponding description of the *AC* semantics [19]. For our purposes it is sufficient to quote from [19] that

- D is an *AC* domain description (ie., a set of propositions);
- R is the set of all sufficiency propositions φ/ψ contained in D;
- for a set Γ of formulas and a set R of sufficiency propositions we define the closure of a set Γ under R, denoted $Cn_R(\Gamma)$, to be the smallest set of formulas such that:
  a) $Cn_R(\Gamma)$ contains Γ;
  b) $Cn_R(\Gamma)$ is logically closed;
  c) for all $\varphi/\psi \in R$, if $\varphi \in Cn_R(\Gamma)$ then $\psi \in Cn_R(\Gamma)$;
  we write $\Gamma \vdash_R \varphi$ to mean that $\varphi \in Cn_R(\Gamma)$;
- an interpretation S of the fluent atoms is a state if Cn(S) is closed under R: $Cn(S) = Cn_R(S)$;
- E(A,S) is the set of fluent formulas φ for which there is an effect proposition

  A **causes** φ **if** ψ

  in D such that S satisfies ψ.

[8] A sufficiency proposition sometimes is abbreviated as φ/ψ.

For simplicity, we do not use executability propositions and say that an action A is prohibited in a state S if there is an effect proposition

`A` **`causes`** `False` **`if`** ψ

in D such that S satisfies ψ. Consequently, a set E = E(A, S) is an explicit effect of A at S if A is not prohibited in S.
Finally, a state $S^/$ may result from doing A in S if there is an explicit effect E of A in S such that

$$Cn\,(S^/) = Cn_R\,[\,(S \cap S^/) \cup E\,] \qquad (1)$$

It is easy to see, that this fixpoint definition is equivalent to the definition 4 for $Res_R(E,S)$ in [15], where R is the set of inference rules, and allows to obtain the same possible next states as the latter. Since there is no “frame/non-frame distinction”, all fluents are subject to inertia. The language DL_{if} was introduced in [14] to overcome this limitation in particular. However, it turns out not to be such a restriction.

3.2 AC¯ Language.

Following the described process of the evolution (fig. 1) in order to introduce *AC¯* we need to augment AC_0 with the concept of a frame and appropriately change the definition (1).
To do that we designate the elements of a certain subset, Φ_{frame}, of the set of frame fluents Φ as inertial. The set Φ_{frame} is called a frame designation for a domain description D if it is not empty, and is included in the augmented language signature $< \Phi, \Phi_{frame}, \Omega >$. The signature $< \Phi, \Omega >$ of the AC_0 language is subsumed by the signature of the *AC¯* language and can be considered as $< \Phi, \varnothing, \Omega >$.
Let L_f denote the set of inertial fluent literals[9]. Then the following fixpoint condition defines a state $S^/$ that may result according to the *AC¯* semantics from doing A in S,

$$Cn\,(S^/) = Cn_R\,[\,(S \cap S^/ \cap L_f\,) \cup E\,] \qquad (2)$$

where E is an explicit effect of A in S as before.
It is interesting to compare, at this point, the condition (2) with the following similar condition

$$S^/ \cap L_f = Cn_R\,[\,(S \cap S^/ \cap L_f\,) \cup E\,] \cap L_f\,, \qquad (3)$$

which basically defines the transition system of the language DL_{if} [14]. It is clear that these two conditions become equivalent if we require that an *AC¯* domain description includes an explicit definition

`always` `F` $\equiv \varphi$

where φ is an inertial fluent formula[10], for each non-inertial fluent. Without this requirement the condition (3) defines possible next states that, in general, do not necessarily satisfy the condition (2). On the other hand, a state $S^/$ satisfying (2) should, obviously, satisfy (3) nevertheless.
In [19] Turner indicated that under the requirement to explicitly define every non-inertial fluent in a domain, a domain description D becomes both an *AC* and a propositional *AR* domain description, and the *AC* models of D are exactly the propositional *AR* models of D. In other words, the *AR*

[9] A literal is *inertial* if the fluent name F occuring in it is inertial [14].
[10] An *inertial fluent formula* is a formula whose atoms are inertial fluents.

Theorem [19] establishes a “horizontal” connection between different branches of the evolutionary tree. In this paper an attempt is made to identify conditions (more precisely, restrictions) which preserve the meaning of domain descriptions produced in the AC_0 language when moving “vertically” to the $AC^{—}$ language and employing the frame concept. Relaxing such restrictions will demonstrate what do we gain by dividing fluents on inertial and non-inertial ones in the framework of an action language with fluent-triggered causality (fact causality).

3.3 A Connection between AC_0 and $AC^{—}$ Language.

As mentioned in [7] domain constraints in the *AR* language “play two different roles: they determine the set of states, and they also determine the indirect effects of actions”. Like the DL_{if} language, the $AC^{—}$ language does not impose the requirement to explicitly define every non-inertial fluent on its domain descriptions. This leaves some of non-inertial fluents less supported as far as indirect effects are concerned. In other words, the “ramification constrains” [15, 19] expressed in the form of sufficiency propositions “φ **suffices for** ψ“ become the only source for updating the truth values of not explicitly defined non-inertial fluents when an action is performed. There is, however, a way to introduce an analogue, albeit not straight-forward one, for an explicit definition adopted in the *AR* and the *AC* languages.

Definition. A fluent F of an $AC^{—}$ (*AC*) domain description D is called an effect-complete[11] fluent if there exist inertial formula φ such that

$$\varphi \textbf{ suffices for } F \in D,$$
$$\neg\varphi \textbf{ suffices for } \neg F \in D,$$

We will denote the set of all effect-complete fluents of a domain description D as Φ_{ec} .
The following proposition establishes a connection between the languages AC_0 and $AC^{—}$ and sheds additional light on the way of producing indirect effects of action according to the semantics of these languages.

Proposition 1. Let $D^{—}$ be an $AC^{—}$ domain description such that each non-inertial fluent is an effect-complete fluent:

$$\Phi \setminus \Phi_{frame} \subseteq \Phi_{ec} .$$

Then $D_0 = D^{—}$ defined in the signature $< \Phi, \emptyset, \Omega >$ by abandoning the frame designation Φ_{frame} is an AC_0 domain description, and the AC_0 models of D_0 are exactly the $AC^{—}$ models of $D^{—}$.

The proof is sketched in the appendix.

The following example enhances the Two-Switches example [11, 15, 19] and illustrates the differences among AC_0 , $AC^{—}$, and DL_{if} dependent on the type of relation between the sets Φ_{ec} and $\Phi \setminus \Phi_{frame}$.

Example 1. There are two switches, a light, and shadows. The light is on only when both switches are on. Light “causes” shadows to appear or, in other words, light is “sufficient” for the appearance of shadows. The AC_0 domain can be described using four propositional fluent names:

[11] The term with a “causal” flavour apparently can be changed to one correlating with “sufficiency” idea.

$$\Phi = \{ \text{Switch1, Switch2, Light, Shadows} \}.$$

There are two action names:

$$\Omega = \{ \text{Toggle1, Toggle2} \}.$$

The propositions are:

```
Toggle1 causes ¬Switch1 if Switch1,            (4)
Toggle1 causes Switch1 if ¬Switch1,            (5)
Toggle2 causes ¬Switch2 if Switch2,            (6)
Toggle2 causes Switch2 if ¬Switch2,            (7)
Switch1 ∧ Switch2 suffices for Light,          (8)
¬Switch1 ∨ ¬Switch2 suffices for ¬Light,       (9)
Light suffices for Shadows,                   (10)
initially ¬Switch1,                           (11)
initially ¬Switch2,                           (12)
initially ¬Light,                             (13)
initially ¬Shadows.                           (14)
```

Consider the action `Toggle2` performed in the initial state S = {`¬Switch1, ¬Switch2, ¬Light, ¬Shadows }.` It is easy to check that the only possible next state satisfying (1) is $S^{/}$ = {`¬Switch1, Switch2, ¬Light, ¬Shadows` } and this, indeed, is the intuitive choice.
Now let us consider the AC^{-} domain description based on (4)-(14) with at least one not effect-complete non-inertial fluent. The frame designation

$$\Phi_{\text{frame}} = \{ \text{Switch1, Switch2} \} \qquad (15)$$

leaves the fluent `Shadows` non-inertial, and propositions (4)-(14) define it as not effect-complete, unlike the fluent `Light` which is non-inertial but effect-complete.
It is worth noting that the AC^{-} domain description (4)-(15) is not an *AC* domain description because the non-inertial fluent `Shadows` is not explicitly defined in terms of inertial fluents. However, this description is a "legal" one in the DL_{if} + QL^{p} language (if we use proper propositions for axioms: **now** φ instead of **initially** φ).
Again consider the action `Toggle2` performed in the initial state S. It turns out that there are no interpretations satisfying the condition (2). The "task" to derive value of the non-inertial fluent `Shadows` fails because this fluent is neither effect-complete nor explicitly defined in terms of inertial fluents. In other words, semantics of the AC^{-} language produces no models for the description (4)-(15).
The DL_{if} + QL^{p} domain description (4)-(15), however, treats two next states as possible - both

$S^{/}_{1}$ = {`¬Switch1, Switch2, ¬Light, ¬Shadows }`
$S^{/}_{2}$ = {`¬Switch1, Switch2, ¬Light, Shadows }`

satisfy the condition (3). In other words, semantics of the DL_{if} language can not decide on the value of the non-inertial fluent `Shadows` and entails two models satisfying `¬Shadows ∨ Shadows.`
Disagreement among models obtained by the AC_{0} , the AC^{-}, and the DL_{if} languages can be avoided either by better frame designation of the domain description including the not effect-

complete fluent `Shadows` in the frame, or by introduction of a “cause” effecting disappearance of “shadows”, eg.

$$\neg\texttt{Light}\ \textbf{suffices for}\ \neg\texttt{Shadows}.$$

The latter way is not always available because, in general, an agent performs reasoning about action in the absence of a complete information. Besides, if we necessarily require from each non-inertial fluent to be effect-complete (or explicitly defined, for that matter) then the provided solution to the ramification problem does not look like a principal one.
Consider the first way (better frame designation) for our example. Let

$$\Phi_{\text{frame}} = \{\ \texttt{Switch1, Switch2, Shadows}\ \} \qquad (16)$$

be a new frame designation. Then the domain description (4)-(14), (16) entails S'_1 as a unique successor state in both the *AC*¯ and the *DL*$_{if}$ semantics.
However, attachment of the not effect-complete fluent `Shadows` to the frame (for not well conceived reason in the first place) does not make reasoning about domain actions more intuitive - as shows, for instance, the action `Toggle2` performed in the state S = {`Switch1, Switch2, Light, Shadows` }. Now all three languages (the AC_0 , the *AC*¯ and the *DL*$_{if}$) agree that the only next possible state must be

$$S'_1 = \{\ \texttt{Switch1},\ \neg\texttt{Switch2},\ \neg\texttt{Light},\ \texttt{Shadows}\ \}.$$

The fluent `Shadows` is in the frame and keeps its value through the change. That should not contradict our intuition - because no one indicated the sufficient reason (the cause - ?) for “shadows” to disappear. However, the designation of the fluent `Shadows` as a non-inertial one would be, perhaps, more intuitive. In any case, the necessity to attach this fluent to the frame arbitrarily seems to be of a concern.

4. Conclusion.

The problem of the persistence of indirect effects as part of the ramification problem was discussed, in particular, in [8]. However, the employment of causal background knowledge might have helped to present a better (a simpler) solution than the one proposed by the *ARD* language. Unfortunately, such a solution imposes, from our point of view, too severe restrictions on a domain description: each non-inertial fluent has to be defined in terms of inertial fluents - either through an explicit definition, or as an effect-complete fluent. This demand decreases the value of the frame concept in a framework of languages with fluent-triggered causality - at least in the considered branch of the evolutionary tree. Probably, further development of action languages will clarify what role should the frame concept play at the intersection of the Frame and the Ramification problems.

Appendix. Proof of proposition 1.

By construction $D_0 = D^{\neg}$. Consider now the set

$$Cn_R\ [\ (S \cap S' \cap L_f\ ) \cup E\ ].$$

It is clear to see that

$$Cn_R\ [\ (S \cap S' \cap L_f\ ) \cup E\ ]\ \subseteq Cn_R\ [\ (S \cap S'\ ) \cup E\ ] \qquad (*)$$

Let L be a set of domain's fluent literals, and L_{ec} be a set of domain's effect-complete fluent literals. Then

$$Cn_R [(S \cap S^{/}) \cup E] = Cn_R [(S \cap S^{/} \cap L) \cup E] =$$
$$Cn_R [\{S \cap S^{/} \cap (L_f \cup L\backslash L_f) \} \cup E] =$$
$$Cn_R [(S \cap S^{/} \cap L_f) \cup \{S \cap S^{/} \cap (L\backslash L_f) \} \cup E].$$

Since $L\backslash L_f \subseteq L_{ec}$, we have

$$Cn_R [(S \cap S^{/}) \cup E] \subseteq Cn_R [(S \cap S^{/} \cap L_f) \cup (S \cap S^{/} \cap L_{ec}) \cup E].$$

Lemma. $\forall \varphi$, if $S \cap S^{/} \cap L_{ec} \vdash_R \varphi$, then $S \cap S^{/} \cap L_f \vdash_R \varphi$.
The proof is trivial.

Using the lemma,

$$Cn_R [(S \cap S^{/}) \cup E] \subseteq Cn_R [(S \cap S^{/} \cap L_f) \cup E] \qquad (**)$$

Expressions (*) and (**) together establish that

$$Cn_R [(S \cap S^{/} \cap L_f) \cup E] = Cn_R [(S \cap S^{/}) \cup E]$$

Therefore the set of states Res(A,S) that may result from doing A at S is the same for the AC_0 and the AC^{-} semantics:

$$Res_0(A,S) = Res^{-} (A,S).$$

Consequently, the AC_0 models of D_0 are exactly the AC^{-} models of D^{-} (see the definition of a model in [19]).

Acknowledgements.
The authors are thankful to members of Sydney Knowledge Systems Group for fruitful discussions on different paradigms of Reasoning about Action.

References:

1. Baker, A.B. (1991). Nonmonotonic Reasoning in the Framework of Situation Calculus, *Artificial Intelligence*, 49, 5 -23.
2. Brewka, G. & Hertzberg, J. (1993). How to Do Things with Worlds: on Formalizing Actions and Plans, *J. Logic Computat.*, Vol. 3, No. 5, 517 532.
3. Fikes, R. & Nilsson, N.J. (1971). STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, *Artificial Intelligence*, 2, 189 - 208.
4. Finger, J.J. (1987). *Exploiting Constraints in Design Synthesis*. PhD Thesis, Stanford University, Stanford, CA.
5. Gelfond, M. & Lifschitz, V. (1993). Representing Action and Change by Logic Programs. *The Journal of Logic Programming*, 17, 301 - 322.
6. Ginsberg, M.L. & Smith, D.E. (1988). Reasoning about Action I: A Possible Worlds Approach, *Artificial Intelligence*, 35, 165 - 195.
7. Giunchiglia E., Kartha, G.N. and V. Lifschitz, V. (1995). Actions with Indirect Effects (Extended Abstract), In *Working Notes of the AAAI Spring Symposium on Extending Theories of Action*, 80-85.
8. Giunchiglia, E. & Lifschitz, V. (1995). Dependent fluents, In *Proceedings of International Joint Conference on Artificial Intelligence*, 1964-1969.

9. Kartha, G.N. (1993). Soundness and Completeness Theorems for Three Formalizations of Action. In *Proceedings of International Joint Conference on Artificial Intelligence*, 724 - 729.
10. Kartha, G.N. (1996). On the Range of Applicability of Baker's Approach to the Frame Problem. In *Proceedings of the Third Symposium on Logical Formalizations of Commonsense Reasoning*.
11. Kartha, G.N. & Lifschitz, V. (1994). Actions with Indirect Effects (Preliminary Report). In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, 341 - 350.
12. Lin, F. (1995). Embracing Causality in Specifying the Indirect Effects of Actions. In *Proceedings of International Joint Conference on Artificial Intelligence*, 1985 - 1991.
13. Lifschitz, V. (1994). *Nested Abnormality Theories*. Manuscript.
14. Lifschitz, V. (1996). Two Components of an Action Language, In *Proceedings of the Third Symposium on Logical Formalizations of Commonsense Reasoning*.
15. McCain, N. & Turner, H. (1995). A causal theory of ramifications and qualifications, In *Proceedings International Joint Conference on Artificial Intelligence*, 1978-1984.
16. McCarthy, J. & Hayes, P. (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Machine Intelligence*, vol. IV, edited by B. Meltzer & D. Michie, 463 - 502.
17. Peppas, P. (1993). *Belief Change and Reasoning about Action. An Axiomatic Approach to modelling Inert Dynamic Worlds and the Connection to the Logic of Theory Change*. PhD thesis, Dept. of Computer Science, University of Sydney.
18. Prokopenko, M., Lindley, C. and Kumar, V.R. (1995). The Application of Reasoning about Action Techniques to Dispatch Management. In *Proceedings of the AI'95 First Australian Workshop on Commonsense Reasoning*, Canberra, 74 -88.
19. Turner, H. (1996). Representing Actions in Default Logic: A Situation Calculus Approach, In *Proceedings of the Third Symposium on Logical Formalizations of Commonsense Reasoning*.