A Logic of Capabilities

(extended abstract)

W. van der Hoek
B. van Linder
J.-J.Ch. Meyer*
Vrije Universiteit Amsterdam
Faculty of Mathematics and Informatics
De Boelelaan 1081a, 1081 HV Amsterdam
The Netherlands
E-mail: wiebe@cs.vu.nl, bernd@cs.vu.nl, jules@cs.vu.nl

January 26, 1993

Abstract

In this paper a logic of capabilities is introduced. We start by defining a language in which not only knowledge and actions of agents can be expressed, but also the abilities. The semantics for this language is given by extended Kripke models. Equivalence transformations on actions are defined in such a way that equivalent actions have equivalent results and that the abilities of an agent are closed under these transformations. The planning capacities of the agent are considered by looking at the Can-predicate and the Cannot-predicate. The former indicates that an agent knows that he can achieve a given goal by performing a given action, the latter indicates that the agent knows that that he cannot achieve the given goal by performing the given action. It turns out that the possibility to express the abilities of an agent provides for a flexible and intuitively appealing framework.

1. Introduction

The planning of an agent in order to achieve some given goal is a topic of much interest in artificial intelligence. An overview of the various aspects of planning can be found in [GL87]. According to [Pol92], in AI planning is considered to be the process of formulating a program of action to achieve some specified goal. Obviously, it is only possible for an agent to achieve a given goal by performing a given sequence of actions if performing this sequence leads to the goal and if it is possible for the agent to perform all the actions in the sequence in the correct order.

The logic that we consider in this paper is based on the concepts of the logic defined in [Moo84]. We follow the notation of [MH], which is based on Harel's dynamic logic (see [Har84]). In this paper we will restrict ourselves to a propositional logic. This is basically done for the sake of simplicity. Nevertheless, in principle an extension to first-order logic is certainly possible and might even be necessary for practical purposes.

We introduce a syntactical operator that can be used to indicate whether a given agent is *capable* of performing a given action. The notion of ability to perform actions bears a slight resemblance to the notion of knowledge of formulae in awareness logic as introduced in [FH88]. In awareness logic an agent needs to be aware of a given formula in order to gain explicit belief of this formula whereas in the logic of capabilities an agent needs to be able to perform a given action to make it possible for him to perform that action. Using capability and knowledge of capability in combination with knowledge of the result of an action provides for some rather intuitive results for the planning capacities of an agent, as can be seen in section 6. With regard to planning, only correctness and feasibility of plans is considered, i.e. our approach does not deal with finding heuristics or any other aspects of planning.

The contents of this paper is as follows. In section 2 the language $\mathcal{L}$ is defined. In section 3.1 a class of models $\mathfrak{M}$, and the semantics for the language $\mathcal{L}$ are defined. In section 3.2, the validity of some formulae from $\mathcal{L}$ for $\mathfrak{M}$ is proved. Furthermore some other properties are shown. In section 3.3 the notion of action transformation is introduced and it is shown that the abilities of an agent are

*also at the Katholieke Universiteit Nijmegen.

closed under the action transformations given here. In section 4 the axioms and proof rules of the logic of capabilities are given. In section 5 the planning capacities of an agent are considered by giving an alternative definition of the so called Can-predicate, originally introduced in [Moo84]. We furthermore define a Cannot-predicate, which also proves to be useful when considering the planning capacities of a rational agent. In section 6 we sum up some properties of both the Can-predicate and the Cannot-predicate. These properties show that both predicates behave in an intuitively acceptable way. Section 7 ends this paper with a discussion and some guidelines for further research.

2. Defining the language $\mathcal{L}$

As stated in section 1, the system defined in this paper is loosely based on the system defined in [Moo84]. The main difference lies in the fact that in Moore's system no attention whatsoever is paid to the physical and mental condition of an agent. It is assumed that whenever an agent knows what some action α is, he is automatically able to perform α, regardless of whether the agent satisfies all physical or mental demands that performance of action α asks from him. In our opinion this approach does not seem to be completely in correspondence with the intuition as can be seen in the following example.

2.1. EXAMPLE. Assume that some agent is planning to win a gold medal in the men's 100 metres at the olympics. To achieve this the agent has to run faster than any other competitor. The agent knows that if he runs fastest he will win the gold medal. He also knows what the action 'run fastest' implies. According to the theory of [Moo84], this suffices to conclude that the agent can win the gold medal. Intuitively this seems to be rather strange: it seems to be necessary that the agent is *physically able* to run fastest; it does not suffice that he knows what this implies.
On the other hand, if the agent knows that if he runs fastest he will win the gold medal and he is able to run fastest, it is rather natural to conclude that the agent can make a successful plan to get a gold medal.

The observation made in example 2.1 clearly shows the need for the introduction of a new operator that can be used to express the ability of an agent to perform a given action. For this the syntactical operator $\mathbf{A}$, indexed with an agent i, is used in $\mathcal{L}$; the expression $\mathbf{A}_i\alpha$ is used to denote the fact that the agent i is able to perform the action α.

2.2. CONVENTION. Throughout this paper we will speak of *actions* to refer to executable codes, programs or algorithms. A typical action is for instance 'open the window'. We will speak of *events* to refer to the execution of some specific action by a specific agent. For instance, the opening of the window by rational agent i is a typical event.

2.3. DEFINITION. The language $\mathcal{L}$ is based on a given set Π of propositional symbols, a finite set At of atomic actions and a finite set $\mathcal{A} = \{1, \ldots, n\}$ of agents. The language $\mathcal{L}$ is defined by the following BNF, where φ is a typical element of the set of formulae $\mathcal{L}$, p is a typical element of the set of propositional symbols Π, α is a typical element of the set of actions Ac, a is a typical element of the set of atomic actions At and i is a typical element of the set of agents $\mathcal{A}$.

$$\begin{array}{ll} \varphi ::= p & | \\ \quad \neg\varphi & | \\ \quad \varphi_1 \wedge \varphi_2 & | \\ \quad \mathbf{K}_i\varphi & | \\ \quad [\mathrm{do}_i(\alpha)]\varphi & | \\ \quad \mathbf{A}_i\alpha & \end{array}$$

where the class Ac of actions is given by:

$$\begin{array}{ll} \alpha ::= a & | \\ \quad \mathtt{skip} & | \\ \quad \mathtt{fail} & | \\ \quad \mathtt{test}\ \varphi & | \\ \quad \alpha_1;\alpha_2 & | \\ \quad \mathtt{if}\ \varphi\ \mathtt{then}\ \alpha_1\ \mathtt{else}\ \alpha_2\ \mathtt{fi} & | \\ \quad \mathtt{while}\ \varphi\ \mathtt{do}\ \alpha_1\ \mathtt{od} & | \\ \quad \alpha_1 + \alpha_2 & \end{array}$$

To provide for optimal flexibility, no demand is made whatsoever upon the use of parentheses in $\mathcal{L}$: whenever one thinks that using parentheses provides for more clarity one is encouraged to use them.

We let p with possible marks denote atomic propositions; $\neg\varphi$ denotes the negation of a formula and $\varphi_1 \wedge \varphi_2$ denotes the conjunction of formulae. The connectives $\vee, \rightarrow$ and $\leftrightarrow$ are defined in the usual way. The formula $\mathbf{tt}$ is introduded as an abbreviation for $p \vee \neg p$ and $\mathbf{ff}$ is assumed to denote $\neg\mathbf{tt}$. The formula $<\mathrm{do}_i(\alpha)>\varphi$ is defined as the dual of $[\mathrm{do}_i(\alpha)]\varphi$: $<\mathrm{do}_i(\alpha)>\varphi \equiv \neg[\mathrm{do}_i(\alpha)]\neg\varphi$.
The formula $\mathbf{K}_i\varphi$ denotes agent i's knowledge of φ. Following common practice in the world of computer science –see for instance [HM85], [FH88] or [MH]– the modal system **S5** is used for the knowledge part of LCap. The axioms for **S5** are given in section 4.

The meaning of the formula $[\mathrm{do}_i(\alpha)]\varphi$ is as follows: if the agent i performs the action α then the formula φ will be

true after the performance of α by i.
The formula $\mathbf{A}_i\alpha$ denotes the ability of agent i to perform the action α. This ability of the agent can be thought of as a combination of physical and mental abilities.

The class of actions Ac can be seen as a *rich-test* extension (see [Har84] or [Har79]) of the class of actions considered in [Hoa69]. The meanings of the respective actions in Ac are subsequently: the atomic action, the empty action, the never succeeding action, a test for the truth of formula φ, sequential composition, conditional composition, repetitive composition and the nondeterministic choice. For a given finite set of agents $\mathcal{A}$ and a given class of actions Ac, the class of events $\text{do}_i(\alpha)$ with $i \in \mathcal{A}$ and $\alpha \in Ac$ is usually denoted by $\mathcal{E}_{\mathcal{A}}^{Ac}$.

The expressive power of language $\mathcal{L}$ is considerable, as can be seen in the following example.

2.4. EXAMPLE (The egocentric agent). Consider the following action: 'If agent i knows that he will feel better after helping his neighbour and he knows that he is able to help him, he will do so, otherwise he will do nothing'. It is possible to denote in the language $\mathcal{L}$ the fact that if agent i is feeling well before performing the egocentric action, he will still feel well or he will feel even better after performing it. Using some obvious abbreviations this formalization is given by:

$$w \rightarrow [\text{do}_i(\texttt{if } \mathbf{K}_i[\text{do}_i(h)]b \wedge \mathbf{K}_i\mathbf{A}_i h \texttt{ then } h \texttt{ else skip fi})](w \vee b)$$

3. Semantics for the language $\mathcal{L}$

In this section a class of models $\mathfrak{M}$ is defined. The models from $\mathfrak{M}$ are used to provide a semantics for the language $\mathcal{L}$. This semantics is an extension of the possible world semantics commonly used to model modal systems. Furthermore the validity of some formulae from $\mathcal{L}$ is proved for the class $\mathfrak{M}$ of models. At the end of this section the notion of action transformation is introduced and equivalences under these transformations are shown.

3.1. Basic definitions

In the following definitions it is assumed that some set $\mathbf{bool} = \{1, 0\}$ of truth values is given. It will always be clear from the context whether 0 and 1 denote truth values or natural numbers.

3.1. DEFINITION.
A Kripke model $\mathcal{M}$ is a tuple $< \mathcal{S}, \pi, \text{R}, \text{r}, \text{c} >$ where

(1) $\mathcal{S}$ is a set of possible worlds.
(2) $\pi : \Pi \times \mathcal{S} \rightarrow \mathbf{bool}$ is a total function that assigns a truth value to propositional symbols in possible worlds.
(3) $\text{R} : \mathcal{A} \rightarrow \wp(\mathcal{S} \times \mathcal{S})$ is the function that yields the accessibility relations for a given agent, i.e. if $(s, s') \in \text{R}(i)$ this means that s' is an epistemic alternative for the agent i in possible world s. Since the knowledge part of the logic of capabilities is given by the modal system **S5**, the function R is such that for all agents i and for all s and s' from $\mathcal{S}$
   - $(s, s) \in \text{R}(i)$
   - $(s, s') \in \text{R}(i) \Rightarrow (s', s) \in \text{R}(i)$
   - $(s, s') \in \text{R}(i), (s', s'') \in \text{R}(i) \Rightarrow (s, s'') \in \text{R}(i)$

   all hold, i.e. $\text{R}(i)$ is an *equivalence relation* (cf. [HM85], [MH]).
(4) $\text{r} : \mathcal{A} \times At \rightarrow \mathcal{S} \rightarrow \wp(\mathcal{S})$ is such that $\text{r}(i, a)(s)$ yields the result of performing atomic action a by agent i in the possible world s. This function is such that for all atomic actions a holds that
   $$\forall i \forall a \forall s[|\text{r}(i, a)(s)| \leq 1]$$
   where $|V|$ denotes the number of elements of the set V.
(5) $\text{c} : \mathcal{A} \times At \rightarrow \mathcal{S} \rightarrow \mathbf{bool}$ is the capability function such that $\text{c}(i, a)(s)$ indicates whether the agent i is capable of performing the action a in the possible world s.

The class of models given by definition 3.1 is denoted by $\mathfrak{M}$.

3.2. DEFINITION (Semantics of the logic of capabilities).
Let $\mathcal{M} =< \mathcal{S}, \pi, \text{R}, \text{r}, \text{c} >$ be some Kripke model from $\mathfrak{M}$. The functions π, r and c are extended by simultanous induction as follows.

$$
\begin{array}{ll}
\pi & : \mathcal{L} \times \mathcal{S} \rightarrow \mathbf{bool} \\
\pi(\neg\varphi, s) & = \neg\pi(\varphi, s) \\
\pi(\varphi \wedge \psi, s) & = \pi(\varphi, s) \wedge \pi(\psi, s) \\
\pi(\mathbf{K}_i\varphi, s) & = \bigwedge\!\!\bigwedge \pi(\varphi, s') \text{ for all } s' \\
 & \quad \text{with } (s, s') \in \text{R}(i). \\
\pi([\text{do}_i(\alpha)]\varphi, s) & = \bigwedge\!\!\bigwedge \pi(\varphi, s') \text{ for all } s' \\
 & \quad \text{with } s' \in \text{r}(i, \alpha)(s). \\
\pi(\mathbf{A}_i\alpha, s) & = \text{c}(i, \alpha)(s)
\end{array}
$$

$$
\begin{array}{ll}
\text{r} & : \mathcal{A} \times Ac \rightarrow \mathcal{S} \rightarrow \wp(\mathcal{S}) \\
\text{r}(i, \texttt{skip})(s) & = \{s\} \\
\text{r}(i, \texttt{fail})(s) & = \emptyset \\
\text{r}(i, \texttt{test } \varphi)(s) & = \{s\} \text{ if } \pi(\varphi, s) = 1
\end{array}
$$

$$\begin{array}{ll}
 & = \emptyset \text{ otherwise} \\
\mathtt{r}(i, \alpha_1; \alpha_2)(s) & = \mathtt{r}(i, \alpha_2)(\mathtt{r}(i, \alpha_1)(s)) \\
\mathtt{r}(i, \mathtt{if}\ \varphi\ \mathtt{then} & = \mathtt{r}(i, \alpha_1)(s) \\
\quad \alpha_1\ \mathtt{else}\ \alpha_2\ \mathtt{fi})(s) & \quad \text{if } \pi(\varphi, s) = 1 \\
 & = \mathtt{r}(i, \alpha_2)(s) \text{ otherwise} \\
\mathtt{r}(i, \mathtt{while}\ \varphi\ \text{do}\ \alpha_1\ \text{od})(s) & = \{s' \mid \exists k \exists s_0 \ldots \exists s_k \\
 & \quad [s_0 = s \wedge s_k = s' \wedge \\
 & \quad \forall j < k[s_{j+1} \in \mathtt{r}(i, \alpha_1)(s_j) \\
 & \quad \wedge \pi(\varphi, s_j) = 1] \\
 & \quad \wedge \pi(\neg\varphi, s') = 1]\} \\
\mathtt{r}(i, \alpha_1 + \alpha_2)(s) & = \mathtt{r}(i, \alpha_1)(s) \cup \mathtt{r}(i, \alpha_2)(s) \\
\text{where } \mathtt{r}(i, \alpha)(\mathcal{S}') & = \bigcup_{s' \in \mathcal{S}'} \mathtt{r}(i, \alpha)(s') \\
 & \quad \text{for } \mathcal{S}' \subseteq \mathcal{S}.
\end{array}$$

$$\begin{array}{ll}
\mathtt{c} & : \mathcal{A} \times Ac \rightarrow \mathcal{S} \rightarrow \mathbf{bool} \\
\mathtt{c}(i, \mathtt{skip})(s) & = 1 \\
\mathtt{c}(i, \mathtt{fail})(s) & = 0 \\
\mathtt{c}(i, \mathtt{test}\ \varphi)(s) & = \pi(\varphi, s) \\
\mathtt{c}(i, \alpha_1; \alpha_2)(s) & = \mathtt{c}(i, \alpha_1)(s) \wedge \\
 & \quad \mathtt{c}(i, \alpha_2)(\mathtt{r}(i, \alpha_1)(s)) \\
\mathtt{c}(i, \mathtt{if}\ \varphi\ \mathtt{then} & = \mathtt{c}(i, \mathtt{test}\ \varphi; \alpha_1)(s) \vee \\
\quad \alpha_1\ \mathtt{else}\ \alpha_2\ \mathtt{fi})(s) & \quad \mathtt{c}(i, \mathtt{test}\ \neg\varphi; \alpha_2)(s) \\
\mathtt{c}(i, \mathtt{while}\ \varphi\ \text{do}\ \alpha_1\ \text{od})(s) & = 1 \text{ if} \\
 & \quad \exists s' \exists k \exists s_0 \ldots \exists s_k [s_0 = s \wedge \\
 & \quad s_k = s' \wedge \forall j < k \\
 & \quad [s_{j+1} \in \mathtt{r}(i, \alpha_1)(s_j) \wedge \\
 & \quad \mathtt{c}(i, \mathtt{test}\ \varphi; \alpha_1)(s_j) = 1] \\
 & \quad \wedge \mathtt{c}(i, \mathtt{test}\ \neg\varphi)(s') = 1] \\
 & = 0 \text{ otherwise} \\
\mathtt{c}(i, \alpha_1 + \alpha_2)(s) & = \mathtt{c}(i, \alpha_1)(s) \vee \mathtt{c}(i, \alpha_2)(s) \\
\text{where } \mathtt{c}(i, \alpha)(\mathcal{S}') & = \bigwedge_{s' \in \mathcal{S}'} \mathtt{c}(i, \alpha)(s) \\
 & \quad \text{for } \mathcal{S}' \subseteq \mathcal{S}.
\end{array}$$

3.3. REMARK. The function π as given in the definitions 3.1 and 3.2 indicates for a given formula and a given possible world whether or not the formula holds in the possible world. The domain and codomain of the function $\mathtt{r}$ are meant to formalize the intuition that $\mathtt{r}$ determines for a given agent i, action α and possible world s, the worlds that are a possible result of the occurrence of the event $\text{do}_i(\alpha)$ in s. The function $\mathtt{c}$ indicates for a given agent, action and possible world whether or not the agent is capable of performing the action in the possible world. One could also think of alternative definitions for the domain and codomain of π, $\mathtt{r}$ and $\mathtt{c}$, for instance $\pi : \mathcal{L} \rightarrow \wp(\mathcal{S})$, $\mathtt{r} : \mathcal{A} \times Ac \rightarrow \wp(\mathcal{S} \times \mathcal{S})$ and $\mathtt{c} : \mathcal{A} \times Ac \rightarrow \wp(\mathcal{S})$. For example the function $\mathtt{c}$ would then yield, for a given agent and a given action, the set of possible worlds in which the agent is capable of performing the action. Obviously, both definitions denote an intuitively identical concept.

3.4. REMARK. Singleton sets $\{s\}$ with $s \in \mathcal{S}$ are usually denoted by s.

3.5. REMARK. The definitions of $\mathtt{r}(i, \mathtt{test}\ \varphi)(s)$ and $\mathtt{c}(i, \mathtt{test}\ \varphi)(s)$ as defined above are based on the idea that a test for φ is actually an action that searches for confirmation of φ; if it is not possible to get confirmation for φ, $\mathtt{test}\ \varphi$ fails. This is in fact the approach taken in dynamic logic. One could also think of a different sort of tests, namely tests that are *informative*. If $\mathtt{test}\ \varphi$ is informative with respect to φ this means that after executing $\mathtt{test}\ \varphi$, the agent either knows that φ holds or knows that $\neg\varphi$ holds. The latter approach could be formalized by replacing $\pi(\varphi, s)$ by $\pi(\mathbf{K}_i\varphi \vee \mathbf{K}_i\neg\varphi, s)$ in the definition of $\mathtt{r}(i, \mathtt{test}\ \varphi)(s)$ and $\mathtt{c}(i, \mathtt{test}\ \varphi)(s)$. For the moment we have chosen to define $\mathtt{r}$ and $\mathtt{c}$ for tests as done above, i.e. analogously to dynamic logic. Nevertheless alternative definitions, possibly based on the assumption that tests are informative, certainly deserve further attention.

3.6. DEFINITION. For all models $\mathcal{M} =< \mathcal{S}, \pi, \mathrm{R}, \mathtt{r}, \mathtt{c} >$ from $\mathfrak{M}$, for all possible worlds $s \in \mathcal{S}$, and for all formulae φ we define:

- $\mathcal{M}, s \models \varphi$ iff $\pi(\varphi, s) = 1$.
- The formula φ is valid in $\mathcal{M}$, notation $\mathcal{M} \models \varphi$, iff $\mathcal{M}, s \models \varphi$ for all $s \in \mathcal{S}$.

3.7. DEFINITION. Let $\mathfrak{M}$ be some class of models. For all formulae φ we define:

- The formula φ is satisfiable in $\mathfrak{M}$ if some Kripke model $\mathcal{M} =< \mathcal{S}, \pi, \mathrm{R}, \mathtt{r}, \mathtt{c} > \in \mathfrak{M}$ and $s \in \mathcal{S}$ exist such that $\mathcal{M}, s \models \varphi$.
- The formula φ is valid in $\mathfrak{M}$, notation $\models \varphi$ or $\models_{\mathfrak{M}} \varphi$, iff $\mathcal{M} \models \varphi$ for all $\mathcal{M} \in \mathfrak{M}$.

The following lemma states some properties that will be useful in proving the theorems of the following sections.

3.8. LEMMA. *For all models $\mathcal{M}$, for all possible worlds $s \in \mathcal{S}$, for all formulae φ and ψ and for all actions α we have:*

- $\mathcal{M}, s \models \neg\varphi \Leftrightarrow \text{not}\ (\mathcal{M}, s \models \varphi)$
- $\mathcal{M}, s \models \varphi \wedge \psi \Leftrightarrow \mathcal{M}, s \models \varphi$ and $\mathcal{M}, s \models \psi$
- $\mathcal{M}, s \models \mathbf{K}_i\varphi \Leftrightarrow \forall s'[(s, s') \in \mathrm{R}(i) \Rightarrow \mathcal{M}, s' \models \varphi]$
- $\mathcal{M}, s \models [\text{do}_i(\alpha)]\varphi \Leftrightarrow$ $\forall s'[s' \in \mathtt{r}(i, \alpha)(s) \Rightarrow \mathcal{M}, s' \models \varphi]$
- $\mathcal{M}, s \models <\text{do}_i(\alpha)> \varphi \Leftrightarrow$ $\exists s'[s' \in \mathtt{r}(i, \alpha)(s) \wedge \mathcal{M}, s' \models \varphi]$

- $\mathcal{M}, s \models \mathbf{A}_i\alpha \Leftrightarrow \mathtt{c}(i, \alpha)(s) = 1$

3.9. REMARK. The definitions of both $\mathtt{r}$ and $\mathtt{c}$ for repetitive composition are constructive ones, and are based on a definition given in [HR83]. Our definition should be compared to that of [Moo84], in which only a fixed-point equation is given. As is shown below, a fixed-point equation on its own is not sufficient.

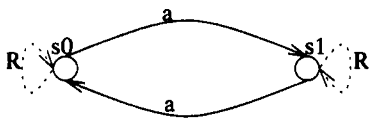
3.10. DEFINITION. Functions $\mathtt{r}'(i, \mathtt{while}\ \varphi\ \mathrm{do}\ \alpha_1\ \mathrm{od})(s)$ and $\mathtt{c}'(i, \mathtt{while}\ \varphi\ \mathrm{do}\ \alpha_1\ \mathrm{od})(s)$ are defined by:

- $\mathtt{r}'(i, \mathtt{while}\ \varphi\ \mathrm{do}\ \alpha_1\ \mathrm{od})(s) =$
  $\mathtt{r}'(i, \mathtt{if}\ \varphi\ \mathtt{then}\ \alpha_1; \mathtt{while}\ \varphi\ \mathrm{do}\ \alpha_1\ \mathrm{od}\ \mathtt{else}\ \mathtt{skip}\ \mathtt{fi})(s)$
- $\mathtt{c}'(i, \mathtt{while}\ \varphi\ \mathrm{do}\ \alpha_1\ \mathrm{od})(s) =$
  $\mathtt{c}'(i, \mathtt{if}\ \varphi\ \mathtt{then}\ \alpha_1; \mathtt{while}\ \varphi\ \mathrm{do}\ \alpha_1\ \mathrm{od}\ \mathtt{else}\ \mathtt{skip}\ \mathtt{fi})(s)$

Assume that $\mathtt{r}'$ and $\mathtt{c}'$ are defined analogously to $\mathtt{r}$ and $\mathtt{c}$ with the exception of the definition for repetitive composition. Then both $\mathtt{r}'$ and $\mathtt{c}'$ are partial functions.

3.11. EXAMPLE. Assume that the set of propositional symbols Π is given by $\Pi = \{p\}$, the set of atomic actions is $\{a\}$ and there is only one agent.
Let the Kripke model $\mathcal{M} =< \mathcal{S}, \pi, \mathrm{R}, \mathtt{r}, \mathtt{c} >$ be such that:

(1) $\mathcal{S} = \{s_0, s_1\}$
(2) $\pi(p, s_0) = \pi(p, s_1) = 1$
(3) R: see the dotted lines in the picture below.
(4) $\mathtt{r}$: see the solid lines in the picture below.
(5) $\mathtt{c}(1, a)(s_0) = \mathtt{c}(1, a)(s_1) = 1$



We find that the computations of both $\mathtt{r}'(1, \mathtt{while}\ p\ \mathrm{do}\ a\ \mathrm{od})(s_0)$ and $\mathtt{c}'(1, \mathtt{while}\ p\ \mathrm{do}\ a\ \mathrm{od})(s_0)$ do not terminate. Note that using definition 3.2 no problems concerning non-termination occur:

- $\mathtt{r}(1, \mathtt{while}\ p\ \mathrm{do}\ a\ \mathrm{od})(s_0) =$
  $\mathtt{r}(1, \mathtt{while}\ p\ \mathrm{do}\ a\ \mathrm{od})(s_1) = \emptyset$
- $\mathtt{c}(1, \mathtt{while}\ p\ \mathrm{do}\ a\ \mathrm{od})(s_0) =$
  $\mathtt{c}(1, \mathtt{while}\ p\ \mathrm{do}\ a\ \mathrm{od})(s_1) = 0$

Using our definition, the equation of definition 3.10 taken as a proposition actually holds, i.e. we have for all s:

- $\mathtt{r}(i, \mathtt{while}\ \varphi\ \mathrm{do}\ \alpha_1\ \mathrm{od})(s) =$
  $\mathtt{r}(i, \mathtt{if}\ \varphi\ \mathtt{then}\ \alpha; \mathtt{while}\ \varphi\ \mathrm{do}\ \alpha_1\ \mathrm{od}\ \mathtt{else}\ \mathtt{skip}\ \mathtt{fi})(s)$
- $\mathtt{c}(i, \mathtt{while}\ \varphi\ \mathrm{do}\ \alpha_1\ \mathrm{od})(s) =$
  $\mathtt{c}(i, \mathtt{if}\ \varphi\ \mathtt{then}\ \alpha; \mathtt{while}\ \varphi\ \mathrm{do}\ \alpha_1\ \mathrm{od}\ \mathtt{else}\ \mathtt{skip}\ \mathtt{fi})(s)$

The notion of determinism is defined as usual.

3.12. DEFINITION (Determinism of events).
Let $\mathcal{M} =< \mathcal{S}, \pi, \mathrm{R}, \mathtt{r}, \mathtt{c} >$ be some Kripke model from $\mathfrak{M}$ and let $\mathrm{do}_i(\alpha)$ be some event from $\mathcal{E}_{\mathcal{A}}^{Ac}$.

- The event $\mathrm{do}_i(\alpha)$ is deterministic in some possible world s from $\mathcal{S}$ if and only if

$$|\mathtt{r}(i, \alpha)(s)| \leq 1$$

- The event $\mathrm{do}_i(\alpha)$ is deterministic for $\mathcal{M}$ if and only if $\mathrm{do}_i(\alpha)$ is deterministic in all s from $\mathcal{S}$.

In definition 3.13 some possible properties dealing with performance of actions by agents and the ability of agents are defined.

3.13. DEFINITION. Let $\mathcal{M} =< \mathcal{S}, \pi, \mathrm{R}, \mathtt{r}, \mathtt{c} >$ be some Kripke model from $\mathfrak{M}$ and let α be some action from Ac.

- The action α is $\mathbf{A}_i$*-realizable* in some possible world $s \in \mathcal{S}$ if and only if

$$\mathcal{M}, s \models \mathbf{A}_i\alpha \rightarrow <\mathrm{do}_i(\alpha)> \mathtt{tt}$$

  holds.
- The action α is A-realizable in some possible world $s \in \mathcal{S}$ if and only if α is $\mathbf{A}_i$-realizable in s for all $i \in \mathcal{A}$.
- The event $\mathrm{do}_i(\alpha)$ is *accordant* in some possible world $s \in \mathcal{S}$ if and only if

$$\mathcal{M}, s \models \mathbf{K}_i[\mathrm{do}_i(\alpha)]\varphi \rightarrow [\mathrm{do}_i(\alpha)]\mathbf{K}_i\varphi$$

  holds for all formulae φ.
- The action α is A-realizable for $\mathcal{M}$ if and only if α is A-realizable in every s from $\mathcal{S}$.
- The event $\mathrm{do}_i(\alpha)$ is accordant for $\mathcal{M}$ if and only if $\mathrm{do}_i(\alpha)$ is accordant in every s from $\mathcal{S}$.

Definition 3.13 states that an event $\mathrm{do}_i(\alpha)$ is accordant, if it holds for arbitrary formula φ that if agent i knows that φ will hold after he performs action α, then agent i will know after performing α that φ holds. $\mathbf{A}_i$-realizability –and hence A-realizability– is a rather important feature of actions when considered from the point of view of planning. If some action α is $\mathbf{A}_i$-realizable, this implies that whenever agent i is capable of performing α, it is possible for him to perform α.

3.2. Validity of formulae from $\mathcal{L}$

Theorems 3.14 and 3.15 deal with the validity of some formulae from $\mathcal{L}$ for the class of models $\mathfrak{M}$ defined in section 3. Theorem 3.14 shows some properties of events, theorem 3.15 shows some properties of capabilities.

3.14. THEOREM. *For all agents i, actions α, α_1 and α_2, and for all formulae φ and ψ we have:*

(1) *For deterministic events* $\mathrm{do}_i(\alpha)$:
$\models <\mathrm{do}_i(\alpha)> \varphi \rightarrow [\mathrm{do}_i(\alpha)]\varphi$

(2) $\models \varphi \leftrightarrow [\mathrm{do}_i(\mathtt{skip})]\varphi$

(3) $\models [\mathrm{do}_i(\mathtt{fail})]\mathbf{ff}$

(4) $\models [\mathrm{do}_i(\mathtt{test}\ \varphi)]\varphi$

(5) $\models [\mathrm{do}_i(\alpha_1;\alpha_2)]\varphi \leftrightarrow [\mathrm{do}_i(\alpha_1)]([\mathrm{do}_i(\alpha_2)]\varphi)$

(6) $\models \varphi \wedge [\mathrm{do}_i(\mathtt{if}\ \varphi\ \mathtt{then}\ \alpha_1\ \mathtt{else}\ \alpha_2\ \mathtt{fi})]\psi$
$\leftrightarrow \varphi \wedge [\mathrm{do}_i(\alpha_1)]\psi$

(7) $\models \neg\varphi \wedge [\mathrm{do}_i(\mathtt{if}\ \varphi\ \mathtt{then}\ \alpha_1\ \mathtt{else}\ \alpha_2\ \mathtt{fi})]\psi$
$\leftrightarrow \neg\varphi \wedge [\mathrm{do}_i(\alpha_2)]\psi$

(8) $\models [\mathrm{do}_i(\mathtt{while}\ \varphi\ \mathtt{do}\ \alpha_1\ \mathtt{od})]\neg\varphi$

(9) $\models [\mathrm{do}_i(\alpha_1 + \alpha_2)]\varphi \leftrightarrow [\mathrm{do}_i(\alpha_1)]\varphi \wedge [\mathrm{do}_i(\alpha_2)]\varphi$

3.15. THEOREM. *For all agents i, for all actions α_1 and α_2 and for all formulae φ we have:*

(1) $\models \mathbf{A}_i\mathtt{skip}$

(2) $\models \neg\mathbf{A}_i\mathtt{fail}$

(3) $\models \varphi \leftrightarrow \mathbf{A}_i\mathtt{test}\ \varphi$

(4) $\models \mathbf{A}_i\mathtt{test}\ \varphi \leftrightarrow \neg\mathbf{A}_i\mathtt{test}\ \neg\varphi$

(5) $\models \mathbf{A}_i(\alpha_1;\alpha_2) \leftrightarrow \mathbf{A}_i\alpha_1 \wedge [\mathrm{do}_i(\alpha_1)]\mathbf{A}_i\alpha_2$

(6) $\models \varphi \wedge \mathbf{A}_i(\mathtt{if}\ \varphi\ \mathtt{then}\ \alpha_1\ \mathtt{else}\ \alpha_2\ \mathtt{fi}) \leftrightarrow$
$\varphi \wedge \mathbf{A}_i\alpha_1$

(7) $\models \neg\varphi \wedge \mathbf{A}_i(\mathtt{if}\ \varphi\ \mathtt{then}\ \alpha_1\ \mathtt{else}\ \alpha_2\ \mathtt{fi}) \leftrightarrow$
$\neg\varphi \wedge \mathbf{A}_i\alpha_2$

(8) $\models \mathbf{A}_i(\alpha_1 + \alpha_2) \leftrightarrow \mathbf{A}_i\alpha_1 \vee \mathbf{A}_i\alpha_2$

Case 2 of theorem 3.15 states intuitively that no agent is capable of failing successfully.

Both for A-realizability and accordancy as defined in section 3.1, correspondences exist. The notion of correspondence is given by the following definition, taken from [Hoe92]. (See [Ben84] for more details on correspondence theory.)

3.16. DEFINITION. A frame $\mathcal{F}$ is a Kripke model without valuation π:

$$\mathcal{F} = < \mathcal{S}, \mathrm{R}, \mathrm{r}, \mathrm{c} >$$

We write $\mathcal{F} \models \varphi$ if and only if $< \mathcal{F}, \pi > \models \varphi$ for all π. If ϑ is any –usually first-order definable– property of $\mathcal{F}$ we say that formula φ corresponds with ϑ, notation $\varphi \sim_{\mathrm{co}} \vartheta$, if for all frames $\mathcal{F}$, $\mathcal{F} \models \varphi \Leftrightarrow \mathcal{F}$ satisfies ϑ.

3.17. THEOREM (Correspondences).
The following correspondences hold:

(1) $\mathbf{A}_i\alpha \rightarrow <\mathrm{do}_i(\alpha)> \mathbf{tt}\ \sim_{\mathrm{co}}$
$\forall s[\mathrm{c}(i,\alpha)(s) = 1 \Rightarrow \mathrm{r}(i,\alpha)(s) \neq \emptyset]$

(2) $\mathbf{K}_i[\mathrm{do}_i(\alpha)]\varphi \rightarrow [\mathrm{do}_i(\alpha)]\mathbf{K}_i\varphi\ \sim_{\mathrm{co}}$
$\forall s_0 \forall s_1[\exists s_2[s_2 \in \mathrm{r}(i,\alpha)(s_0) \wedge (s_2, s_1) \in \mathrm{R}(i)] \Rightarrow$
$\exists s_3[(s_0, s_3) \in \mathrm{R}(i) \wedge s_1 \in \mathrm{r}(i,\alpha)(s_3)]]$

It turns out that for some given model A-realizability of all actions from Ac can easily be achieved, by imposing a restriction on the functions r and c when applied to atomic actions.

3.18. THEOREM. *For all models $\mathcal{M} =< \mathcal{S}, \pi, \mathrm{R}, \mathrm{r}, \mathrm{c} >$ from $\mathfrak{M}$ we have:*

$$(\forall s \forall i \forall a \in At[\mathrm{c}(i,a)(s) = 1 \Rightarrow \mathrm{r}(i,a)(s) \neq \emptyset]) \Rightarrow$$
$$(\forall s \forall i \forall \alpha \in Ac[\mathrm{c}(i,\alpha)(s) = 1 \Rightarrow \mathrm{r}(i,\alpha)(s) \neq \emptyset])$$

3.19. REMARK. It turns out that A-realizability is a very desirable property. Even stronger, models that do not have the property that all occurring actions are A-realizable provide for rather counterintuitive results. For instance, suppose that in some given possible world s an agent is capable of performing the action α but no possible world s' exists such that $s' \in \mathrm{r}(i,\alpha)(s)$. Then it is not possible for the agent to execute α. Furthermore it follows that the agent is capable of performing the action $\alpha;\mathtt{fail}$ in s.
To avoid any counterintuitive situations we demand from now on that all occurring Kripke models are such that all actions are A-realizable, i.e. all models $< \mathcal{S}, \pi, \mathrm{R}, \mathrm{r}, \mathrm{c} >$ are such that $\forall a \in At[\mathrm{c}(i,a)(s) = 1 \Rightarrow \mathrm{r}(i,a)(s) \neq \emptyset]$ holds for all i and s; these models are said to meet the *realizability condition.*

3.20. REMARK. Note that the realizability condition does not imply that $< \mathrm{do}_i(\alpha) > \mathbf{tt} \rightarrow \mathbf{A}_i\alpha$ holds, i.e. $< \mathrm{do}_i(\alpha) > \mathbf{tt}$ and $\mathbf{A}_i\alpha$ are not equivalent. To let our framework be as general and flexible as possible, we allow for the situation in which both $\neg\mathbf{A}_i\alpha$ and $<\mathrm{do}_i(\alpha)> \mathbf{tt}$ hold in a given possible world. Intuitively, this possible world represents a situation in which execution of α would lead to the truth of the formula φ, but since the agent is not capable of performing α, the event $\mathrm{do}_i(\alpha)$ will not occur.

3.3. Transformations on actions

The notion of *action transformation* is a well-known one in the theory of software engineering. In software engineering, action transformations are used to transform programs in other, probably more efficient or better understandable, programs. (For more information on action transformations and transformational programming see [Par90].)
We define two actions to be (action)-equivalent, if they can be transformed into one another by means of a finite number of applications of action transformations. In theorem 3.23 below, it is shown that our semantics is such that the results upon termination of two equivalent actions are logically equivalent, and furthermore that the abilities of an agent are closed under some well-known transformations.
The following definition gives the action transformations that we will take into consideration. Some of these are slightly modified versions of the transformations given in [Par90].

3.21. DEFINITION. The one-step action transformations $\sim_a$ are defined as follows.

(1) $\texttt{skip}; \alpha \sim_a \alpha$

(2) $\alpha; \texttt{skip} \sim_a \alpha$

(3) $\texttt{fail}; \alpha \sim_a \texttt{fail}$

(4) $\alpha; \texttt{fail} \sim_a \texttt{fail}$

(5) $\texttt{test}\ \varphi \sim_a \texttt{if}\ \varphi\ \texttt{then skip else fail fi}$

(6) $\texttt{test}\ \varphi; \texttt{test}\ \varphi \sim_a \texttt{test}\ \varphi$

(7) $\alpha_1; (\alpha_2; \alpha_3) \sim_a (\alpha_1; \alpha_2); \alpha_3$

(8) $\texttt{if tt then}\ \alpha_1\ \texttt{else}\ \alpha_2\ \texttt{fi} \sim_a \alpha_1$

(9) $\texttt{if ff then}\ \alpha_1\ \texttt{else}\ \alpha_2\ \texttt{fi} \sim_a \alpha_2$

(10) $\texttt{if}\ \varphi\ \texttt{then}\ \alpha_1\ \texttt{else}\ \alpha_1\ \texttt{fi} \sim_a \alpha_1$

(11) $\texttt{if}\ \varphi\ \texttt{then}\ \alpha_1\ \texttt{else}\ \alpha_2\ \texttt{fi} \sim_a$
$\texttt{if}\ \neg\varphi\ \texttt{then}\ \alpha_2\ \texttt{else}\ \alpha_1\ \texttt{fi}$

(12) $\texttt{if}\ \varphi\ \texttt{then if}\ \varphi\ \texttt{then}\ \alpha_1\ \texttt{else}\ \alpha_2\ \texttt{fi else}\ \alpha_2\ \texttt{fi} \sim_a$
$\texttt{if}\ \varphi\ \texttt{then}\ \alpha_1\ \texttt{else}\ \alpha_2\ \texttt{fi}$

(13) $\texttt{while}\ \varphi\ \texttt{do}\ \alpha_1\ \texttt{od} \sim_a$
$\texttt{if}\ \varphi\ \texttt{then}\ \alpha_1; \texttt{while}\ \varphi\ \texttt{do}\ \alpha_1\ \texttt{od else skip fi}$

(14) $\texttt{if}\ \varphi\ \texttt{then}\ \alpha_1\ \texttt{else}\ \alpha_2\ \texttt{fi} \sim_a$
$\texttt{if}\ \varphi\ \texttt{then}\ \alpha_1\ \texttt{else fail fi} +$
$\texttt{if}\ \neg\varphi\ \texttt{then}\ \alpha_2\ \texttt{else fail fi}$

(15) $\texttt{if}\ \varphi\ \texttt{then}\ \alpha_1\ \texttt{else}\ \alpha_2\ \texttt{fi} \sim_a$
$\texttt{test}\ \varphi; \alpha_1 + \texttt{test}\ \neg\varphi; \alpha_2$

(16) $\alpha + \alpha \sim_a \alpha$

(17) $\alpha_1 + \alpha_2 \sim_a \alpha_2 + \alpha_1$

(18) $\alpha_1 + (\alpha_2 + \alpha_3) \sim_a (\alpha_1 + \alpha_2) + \alpha_3$

The actions α and α' from Ac are *a-equivalent* (for action-equivalent), notation $\alpha \equiv_a \alpha'$, if and only if action α can be transformed in α' by application of a finite number of $\sim_a$ steps.

3.22. REMARK. To obtain equivalences under the second transformation concerning the fail-statement (transformation 4), it is used essentially that all occurring models satisfy the realizability condition. If the realizability condition is not met, the capabilities of an agent are not closed under this transformation.

Using definition 3.21 we find the following to hold.

3.23. THEOREM. *For all the actions α and α' from Ac, for all the agents i from $\mathcal{A}$, and for all the formulae φ we have:*

- $\alpha \equiv_a \alpha' \Rightarrow \models [\mathrm{do}_i(\alpha)]\varphi \leftrightarrow [\mathrm{do}_i(\alpha')]\varphi$
- $\alpha \equiv_a \alpha' \Rightarrow \models \mathbf{A}_i\alpha \leftrightarrow \mathbf{A}_i\alpha'$

Theorem 3.23 states that if two actions are equivalent, these actions have the same result upon performing and furthermore that if an agent is able to perform one of these actions he is also able to perform the other one.

4. Defining LCap

The logic LCap is defined by the following axioms and proof rules.

THE AXIOMS:

A1. All propositional tautologies and their epistemic and dynamic instances.

A2. $(\mathbf{K}_i\varphi \wedge \mathbf{K}_i(\varphi \to \psi)) \to \mathbf{K}_i\psi$
A3. $\mathbf{K}_i\varphi \to \varphi$
A4. $\mathbf{K}_i\varphi \to \mathbf{K}_i\mathbf{K}_i\varphi$
A5. $\neg\mathbf{K}_i\varphi \to \mathbf{K}_i\neg\mathbf{K}_i\varphi$

Ac0. $([\mathrm{do}_i(\alpha)](\varphi \to \psi) \wedge [\mathrm{do}_i(\alpha)]\varphi) \to [\mathrm{do}_i(\alpha)]\psi$
Ac1. $[\mathrm{do}_i(\texttt{skip})]\varphi \leftrightarrow \varphi$
Ac2. $[\mathrm{do}_i(\texttt{fail})]\mathbf{ff}$
Ac3. $[\mathrm{do}_i(\texttt{test}\ \varphi)]\psi \leftrightarrow (\varphi \to \psi)$
Ac4. $[\mathrm{do}_i(\alpha_1; \alpha_2)]\varphi \leftrightarrow [\mathrm{do}_i(\alpha_1)][\mathrm{do}_i(\alpha_2)]\varphi$
Ac5. $[\mathrm{do}_i(\texttt{if}\ \varphi\ \texttt{then}\ \alpha_1\ \texttt{else}\ \alpha_2\ \texttt{fi})]\psi \leftrightarrow$
$(\varphi \to [\mathrm{do}_i(\alpha_1)]\psi) \wedge (\neg\varphi \to [\mathrm{do}_i(\alpha_2)]\psi)$
Ac6. $[\mathrm{do}_i(\texttt{while}\ \varphi\ \texttt{do}\ \alpha_1\ \texttt{od})]\psi \leftrightarrow ((\neg\varphi \wedge \psi) \vee$
$(\varphi \wedge [\mathrm{do}_i(\alpha_1)][\mathrm{do}_i(\texttt{while}\ \varphi\ \texttt{do}\ \alpha_1\ \texttt{od})]\psi))$

Ac7. $[\mathrm{do}_i(\alpha_1 + \alpha_2)]\varphi \leftrightarrow [\mathrm{do}_i(\alpha_1)]\varphi \wedge [\mathrm{do}_i(\alpha_2)]\varphi$

Ab1. $\mathbf{A}_i\mathtt{skip}$
Ab2. $\neg\mathbf{A}_i\mathtt{fail}$
Ab3. $\mathbf{A}_i\mathtt{test}\ \varphi \leftrightarrow \varphi$
Ab4. $\mathbf{A}_i(\alpha_1;\alpha_2) \leftrightarrow \mathbf{A}_i\alpha_1 \wedge [\mathrm{do}_i(\alpha_1)]\mathbf{A}_i\alpha_2$
Ab5. $\mathbf{A}_i\mathtt{if}\ \varphi\ \mathtt{then}\ \alpha_1\ \mathtt{else}\ \alpha_2\ \mathtt{fi} \leftrightarrow$
$(\varphi \rightarrow \mathbf{A}_i\alpha_1) \wedge (\neg\varphi \rightarrow \mathbf{A}_i\alpha_2)$
Ab6. $\mathbf{A}_i\mathtt{while}\ \varphi\ \mathrm{do}\ \alpha_1\ \mathrm{od} \leftrightarrow (\neg\varphi \vee$
$(\varphi \wedge \mathbf{A}_i\alpha_1 \wedge [\mathrm{do}_i(\alpha_1)]\mathbf{A}_i\mathtt{while}\ \varphi\ \mathrm{do}\ \alpha_1\ \mathrm{od}))$
Ab7. $\mathbf{A}_i(\alpha_1 + \alpha_2) \leftrightarrow \mathbf{A}_i\alpha_1 \vee \mathbf{A}_i\alpha_2$

THE PROOF RULES:

$$R1.\ \frac{(\varphi \wedge \psi) \rightarrow [\mathrm{do}_i(\alpha_1)]\psi}{\psi \rightarrow [\mathrm{do}_i(\mathtt{while}\ \varphi\ \mathrm{do}\ \alpha_1\ \mathrm{od})]\psi}$$

$$R2.\ \frac{\varphi \rightarrow (\mathbf{A}_i\alpha_1 \wedge <\mathrm{do}_i(\mathtt{while}\ \varphi\ \mathrm{do}\ \alpha_1\ \mathrm{od})>\neg\varphi)}{\mathbf{A}_i\mathtt{while}\ \varphi\ \mathrm{do}\ \alpha_1\ \mathrm{od}}$$

$$R3.\ \frac{\varphi \quad \varphi \rightarrow \psi}{\psi}$$

$$R4.\ \frac{\varphi}{\mathbf{K}_i\varphi}$$

$$R5.\ \frac{\varphi}{[\mathrm{do}_i(\alpha)]\varphi}$$

The following definition introduces the notion of proofs for LCap.

4.1. DEFINITION. A proof of a formula φ is a finite sequence of formulae such that

(1) The formula φ is the last formula in the sequence, and
(2) each formula in the sequence is either an axiom of LCap or follows by one of the inference rules of LCap from previous formulae in the sequence.

If a proof of the formula φ exists, φ is called a theorem of LCap and we write $\vdash_{\mathrm{LCap}} \varphi$, or $\vdash \varphi$ for short.

4.2. THEOREM. *LCap is sound with regard to $\mathfrak{M}$, i.e.*

$$\vdash_{\mathrm{LCap}} \varphi \Rightarrow \models_{\mathfrak{M}} \varphi$$

4.3. REMARK. Although we did not yet deal with proving the completeness of LCap for $\mathfrak{M}$, we do not expect this proof to be substantially more difficult than the proofs of completeness for propositional dynamic logic (see for instance [Har84]).

Using the axioms and proof rules of LCap, it is possible to give the proof-theoretical equivalents of the proofs given in the preceding sections. In particular it is possible to prove the proof-theoretical equivalent of theorem 3.23.

4.4. THEOREM. *For all α and α' from Ac, for all i from $\mathcal{A}$, and for all formulae φ we have*

- $\alpha \equiv_a \alpha' \Rightarrow \vdash [\mathrm{do}_i(\alpha)]\varphi \leftrightarrow [\mathrm{do}_i(\alpha')]\varphi$
- $\alpha \equiv_a \alpha' \Rightarrow \vdash \mathbf{A}_i\alpha \leftrightarrow \mathbf{A}_i\alpha'$

5. The Can-predicate and the Cannot-predicate

In [Moo84] the Can-predicate is –in a slightly different notation– introduced by:

$$\mathbf{Can}_i(\alpha,\varphi) \Leftarrow \exists x[\mathbf{K}_i(x = \alpha \wedge [\mathrm{do}_i(\alpha)]\varphi)]$$

Intuitively, $\mathbf{Can}_i(\alpha,\varphi)$ means that agent i knows how to achieve φ by performing action α. In terms of [Moo84], the agent knows how to perform α since he knows what action α is, and he knows that performing α will lead to the truth of φ.
Notice that in the formula above '$\Leftarrow$' may not be replaced by '$\Leftrightarrow$': in carrying out a complex action –for instance a sequence $\alpha;\beta$– an agent does not need to know a priori what action $\alpha;\beta$ is; it suffices that he knows now what action α is and that he knows after performing α what action β is.
As stated in section 1, we will define the Can-predicate in a different way, taking into account the abilities of the agent. The Can-predicate is introduced as a syntactic concept whose semantics is given by means of the semantics of the parts it consists of.
The intended intuitive meaning of $\mathbf{Can}_i(\alpha,\varphi)$ as we define it is the following:

'Agent i knows that he can achieve φ by performing α.'

Starting from this intuitive meaning, four possible definitions of the Can-predicate can be given, each of these more or less corresponding to the intended intuitive meaning.

(1) $\mathbf{Can}_i(\alpha, \varphi) \equiv [\mathrm{do}_i(\alpha)]\varphi \wedge \mathbf{A}_i\alpha$
(2) $\mathbf{Can}_i(\alpha, \varphi) \equiv [\mathrm{do}_i(\alpha)]\varphi \wedge \mathbf{K}_i\mathbf{A}_i\alpha$
(3) $\mathbf{Can}_i(\alpha, \varphi) \equiv \mathbf{K}_i[\mathrm{do}_i(\alpha)]\varphi \wedge \mathbf{A}_i\alpha$
(4) $\mathbf{Can}_i(\alpha, \varphi) \equiv \mathbf{K}_i[\mathrm{do}_i(\alpha)]\varphi \wedge \mathbf{K}_i\mathbf{A}_i\alpha$

Since we want the agent to *know* that he can achieve φ, it seems natural to demand that the agent *knows* that performing α will lead to φ, i.e. $\mathbf{K}_i[\mathrm{do}_i(\alpha)]\varphi$ seems to be a natural part of the definition of $\mathbf{Can}_i(\alpha, \varphi)$. In particular this means that cases 1 and 2 do not provide for intuitively acceptable definitions of the Can-predicate.
Furthermore it seems reasonable to demand that the agent knows that he is able to perform a given action. If an agent is able to perform some action α but he himself is not aware of this ability, then this action α is obviously not used in any plans of the agent.
This observation leads to the following definition of the Can-predicate.

5.1. DEFINITION. Predicate $\mathbf{Can}_i(\alpha, \varphi)$ is defined by:

$$\mathbf{Can}_i(\alpha, \varphi) \equiv \mathbf{K}_i([\mathrm{do}_i(\alpha)]\varphi \wedge \mathbf{A}_i\alpha)$$

In the spirit of definition 5.1 one could also think of a Cannot-predicate. This predicate is not just the negation of the Can-predicate, but has as its intended meaning that the agent knows that he cannot reach some goal φ by performing some action α, since he knows that either the action does not lead to the desired goal or he is not capable of performing the action.

5.2. DEFINITION. Predicate $\mathbf{Cannot}_i(\alpha, \varphi)$ is defined by:

$$\mathbf{Cannot}_i(\alpha, \varphi) \equiv \mathbf{K}_i(\neg[\mathrm{do}_i(\alpha)]\varphi \vee \neg\mathbf{A}_i\alpha)$$

Using definitions 5.1 and 5.2 it is possible to divide the class of actions, for a given agent and and a given goal, in three parts:

(1) Actions of which the agent knows that he can perform them to achieve the goal.
(2) Actions of which the agent knows that performing them will not lead to the goal.
(3) Actions of which the agent does not know whether they can be used to achieve the goal or not.

6. Properties of the Can-predicate and the Cannot-predicate

In the preceding section the definitions of the Can-predicate and the Cannot-predicate are given. In this section, we sum up some of the properties that these predicates have. In the theorems 6.1 and 6.2, we list the properties of the *means* part of the predicates, this is the α in $\mathbf{Can}_i(\alpha, \varphi)$ and $\mathbf{Cannot}_i(\alpha, \varphi)$. Theorems 6.3 and 6.4 list the properties of the *goal* part of the predicates, the φ in $\mathbf{Can}_i(\alpha, \varphi)$ and $\mathbf{Cannot}_i(\alpha, \varphi)$. In theorem 6.5 two relations that exist between the Can-predicate and the Cannot-predicate are given. It furthermore turns out that both the Can-predicate and the Cannot-predicate are closed under equivalence of actions.

6.1. THEOREM (Properties concerning the means part).
For all agents i, actions α_1 and α_2 and for all formulae φ and ψ the following hold.

(1) $\models \mathbf{Can}_i(\mathtt{skip}, \varphi) \leftrightarrow \mathbf{K}_i\varphi$
(2) $\models \neg\mathbf{Can}_i(\mathtt{fail}, \mathbf{tt})$
(3) $\models \mathbf{Can}_i(\mathtt{test}\ \varphi, \psi) \leftrightarrow \mathbf{K}_i(\varphi \wedge \psi)$
(4) $\models \mathbf{Can}_i(\alpha_1, \mathbf{Can}_i(\alpha_2, \varphi)) \rightarrow \mathbf{Can}_i(\alpha_1; \alpha_2, \varphi)$
(5) *For accordant events* $\mathrm{do}_i(\alpha_1)$*:*
$\models \mathbf{Can}_i(\alpha_1; \alpha_2, \varphi) \rightarrow [\mathrm{do}_i(\alpha_1)]\mathbf{Can}_i(\alpha_2, \varphi)$
(6) $\models \mathbf{Can}_i(\alpha_1 + \alpha_2, \varphi) \rightarrow \mathbf{Can}_i(\alpha_1, \varphi) \vee \mathbf{Can}_i(\alpha_2, \varphi)$

6.2. THEOREM (Properties concerning the means part).
For all agents i, actions α_1 and α_2 and for all formulae φ and ψ the following hold.

(1) $\models \mathbf{Cannot}_i(\mathtt{skip}, \varphi) \leftrightarrow \mathbf{K}_i\neg\varphi$
(2) $\models \mathbf{Cannot}_i(\mathtt{fail}, \mathbf{tt})$
(3) $\models \mathbf{Cannot}_i(\mathtt{test}\ \varphi, \psi) \leftrightarrow \mathbf{K}_i(\varphi \rightarrow \neg\psi)$
(4) $\models \mathbf{Cannot}_i(\alpha_1; \alpha_2, \varphi) \rightarrow$
$\mathbf{Cannot}_i(\alpha_1, \mathbf{Can}_i(\alpha_2, \varphi))$
(5) $\models \mathbf{Cannot}_i(\alpha_1, \varphi) \wedge \mathbf{Cannot}_i(\alpha_2, \varphi) \rightarrow$
$\mathbf{Cannot}_i(\alpha_1 + \alpha_2, \varphi)$

6.3. THEOREM (Properties concerning the goal part).
For all agents i, for all actions α and for all formulae φ, φ_1 and φ_2 the following hold.

(1) $\models \mathbf{Can}_i(\alpha, \varphi_1 \wedge \varphi_2) \leftrightarrow \mathbf{Can}_i(\alpha, \varphi_1) \wedge \mathbf{Can}_i(\alpha, \varphi_2)$
(2) $\models \mathbf{Can}_i(\alpha, \varphi_1) \vee \mathbf{Can}_i(\alpha, \varphi_2) \rightarrow \mathbf{Can}_i(\alpha, \varphi_1 \vee \varphi_2)$
(3) $\models \mathbf{Can}_i(\alpha, \neg\varphi) \rightarrow \neg\mathbf{Can}_i(\alpha, \varphi)$
(4) $\models \mathbf{Can}_i(\alpha, \varphi) \wedge \mathbf{K}_i[\mathrm{do}_i(\alpha)](\varphi \rightarrow \psi) \rightarrow \mathbf{Can}_i(\alpha, \psi)$

6.4. THEOREM (Properties concerning the goal part).
For all agents i, for all actions α and for all formulae φ, φ_1 and φ_2 the following hold.

(1) $\models \mathbf{Cannot}_i(\alpha, \varphi_1) \vee \mathbf{Cannot}_i(\alpha, \varphi_2) \rightarrow$
$\mathbf{Cannot}_i(\alpha, \varphi_1 \wedge \varphi_2)$

(2) $\models \mathbf{Cannot}_i(\alpha, \varphi_1 \vee \varphi_2) \rightarrow \mathbf{Cannot}_i(\alpha, \varphi_1) \wedge \mathbf{Cannot}_i(\alpha, \varphi_2)$

(3) $\models \mathbf{Cannot}_i(\alpha, \varphi) \wedge \mathbf{Cannot}_i(\alpha, \neg\varphi) \leftrightarrow \mathbf{K}_i((<\mathrm{do}_i(\alpha)> \varphi \wedge <\mathrm{do}_i(\alpha)> \neg\varphi) \vee \neg\mathbf{A}_i\alpha)$

(4) $\models \mathbf{Cannot}_i(\alpha, \varphi) \wedge \mathbf{K}_i[\mathrm{do}_i(\alpha)](\psi \rightarrow \varphi) \rightarrow \mathbf{Cannot}_i(\alpha, \psi)$

6.5. THEOREM. *For all agents i, for all actions α and for all formulae φ the following hold.*

(1) $\models \mathbf{Can}_i(\alpha, \varphi) \rightarrow \neg\mathbf{Cannot}_i(\alpha, \varphi)$

(2) $\models \mathbf{Can}_i(\alpha, \varphi) \rightarrow \mathbf{Cannot}_i(\alpha, \neg\varphi)$

From case 4 of theorem 6.3 and case 4 of theorem 6.4, the following corollary can be derived.

6.6. COROLLARY. *The following rules are, for all agents i, actions α and formulae φ and ψ, sound for 𝔐:*

$$\frac{\mathbf{Can}_i(\alpha, \varphi) \quad \varphi \rightarrow \psi}{\mathbf{Can}_i(\alpha, \psi)}$$

$$\frac{\mathbf{Cannot}_i(\alpha, \varphi) \quad \psi \rightarrow \varphi}{\mathbf{Cannot}_i(\alpha, \psi)}$$

The following corollary is a consequence of theorem 3.23.

6.7. COROLLARY. *For all agents i, actions α and α′ and for all formulae φ we have:*

- $\alpha \equiv_a \alpha' \Rightarrow \models \mathbf{Can}_i(\alpha, \varphi) \leftrightarrow \mathbf{Can}_i(\alpha', \varphi)$
- $\alpha \equiv_a \alpha' \Rightarrow \models \mathbf{Cannot}_i(\alpha, \varphi) \leftrightarrow \mathbf{Cannot}_i(\alpha', \varphi)$

7. Discussion

In this paper we introduced a new operator that denotes the capabilities of a rational agent. Combining this operator with a logic that combines knowledge and actions provides for an interesting new approach that can serve as a base for further development.
Our future research regarding the topics introduced in this paper will be focused on the following four points:

- Considering various possible definitions of $\pi(\mathtt{test}\ \varphi, s)$, for nontrivial φ containing events, and the consequences of modifying this definition on the closure of abilities under the action transformations.
- Proving the completeness of LCap for the class of models 𝔐.
- Defining a first-order extension of $\mathcal{L}$ and extending the semantics and the logic LCap to deal with this.
- Introducing defeasible reasoning and investigate planning in uncertain circumstances.

References

[Ben84] J. van Benthem. Correspondence theory. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 2, pages 167–247. Reidel, Dordrecht, 1984.

[FH88] R. Fagin and J.Y. Halpern. Belief, awareness and limited reasoning. *Artificial Intelligence*, 34:39–76, 1988.

[GL87] M.P. Georgeff and A.L. Lansky, editors. *Reasoning about Actions and Plans.* The 1986 Workshop, Morgan Kaufmann, 1987.

[Har79] D. Harel. *First-Order Dynamic Logic*, volume 68 of *LNCS*. Springer-Verlag, 1979.

[Har84] D. Harel. Dynamic logic. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 2, pages 497–604. Reidel, Dordrecht, 1984.

[HM85] J.Y. Halpern and Y.O. Moses. A guide to the modal logics of knowledge and belief. In *Proc. 9th IJCAI*, pages 480–490, 1985.

[Hoa69] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12:576–580, 1969.

[Hoe92] W. van der Hoek. *Modalities for Reasoning about Knowledge and Quantities.* PhD thesis, Vrije Universiteit Amsterdam, 1992.

[HR83] J. Halpern and J. Reif. The propositional dynamic logic of deterministic, well-structured programs. *Theoretical Computer Science*, 27:127–165, 1983.

[MH] J.-J. Ch. Meyer and W. van der Hoek. Epistemic logic for AI and computer science. Manuscript.

[Moo84] R.C. Moore. A formal theory of knowledge and action. Technical Report 320, SRI International, 1984.

[Par90] H.A. Partsch. *Specification and Transformation of Programs.* Springer Verlag, 1990.

[Pol92] M.E. Pollack. The uses of plans. *Artificial Intelligence*, 57:43–68, 1992.