Chapitre 12

Raisonnement sur l'action et le changement[1]

F. Dupin de Saint-Cyr (IRIT), A. Herzig (IRIT), J. Lang (LAMSADE) et P. Marquis (CRIL)

Ce chapitre présente un état de la recherche sur la formalisation du raisonnement d'un agent au sujet d'un système dynamique qu'il peut observer (au moins partiellement) et sur lequel il peut éventuellement agir (ou du moins décider d'actions à exécuter, même s'il ne s'en charge pas directement). Nous définissons d'abord les concepts de base du domaine : les états du système, les actions ontiques et épistémiques, les observations ; puis les processus de raisonnement : la prédiction, la progression, la régression, la postdiction, le filtrage, l'abduction et l'extrapolation. Nous décrivons ensuite les problèmes classiques de représentation des actions et comment ces problèmes sont résolus dans certains langages standard de représentation d'actions (ou langages d'actions) ; pour des raisons d'espace, nous nous focalisons sur les langages principaux que sont le calcul des situations, STRIPS et les langages propositionnels d'action, les logiques temporelles, et les réseaux bayésiens dynamiques. Nous abordons enfin un cas particulier de prédiction : la mise-à-jour des connaissances.

12.1 Introduction

Dans ce chapitre, on s'intéresse à la formalisation du raisonnement d'un agent unique (l'agent de référence) qui peut réaliser des observations sur un système dynamique et envisager des actions à exécuter sur celui-ci. Le raisonnement sur l'action et le changement fait partie des premiers problèmes abordés par l'intelligence artificielle (IA) ; il a en particulier fait l'objet de l'article fondateur de McCarthy et Hayes [53]. La recherche dans ce domaine a été très productive jusqu'à la fin des années 1990. Elle a permis, entre autres, de proposer des solutions aux différents problèmes de représentation des actions, d'élaborer une classification des formalismes s'appuyant sur leur expressivité et de progresser vers l'automatisation du raisonnement sur l'action et le changement, en proposant par exemple des algorithmes pour les principaux processus de raisonnement pour divers langages d'actions et/ou en identifiant la complexité algorithmique de ces processus.

Les raisons pour lesquelles un agent peut désirer agir afin de modifier ou mieux connaître l'état dans lequel un système dynamique se trouve sont variées. Par exemple, il peut s'agir de faire évoluer le système vers une configuration que l'agent juge meilleure que sa configuration actuelle, voire vers une configuration optimale pour l'agent (comme déplacer un robot d'un emplacement à un autre). Il peut s'agir aussi de faire en sorte qu'une certaine propriété d'un système dynamique

1. Ce document est une version de travail, le chapitre est publié dans *Panorama de l'intelligence artificielle. Ses bases méthodologiques, ses développements*. Pierre Marquis, Odile Papini, Henri Prade (Eds.), Cépaduès, 12, Vol. 1, 2014

soit maintenue, ou que les états successifs ne s'écartent pas trop d'une trajectoire normale (par exemple, réguler le fonctionnement d'un système physique, comme celui d'un haut-fourneau ou celui d'un patient en réanimation). De tels scénarios font intervenir des concepts (état, action, observation, etc.) et des processus les liant (planification, prédiction, explication, etc.).

Par « formalisation », on entend d'abord modélisation des concepts et des processus-clés qui sont mis en œuvre dans de tels scénarios (il s'agit de les définir rigoureusement d'un point de vue mathématique) puis on s'attache à les représenter (c'est-à-dire à préciser comment l'information est codée) et les automatiser (préciser comment les processus sont réalisés au moyen d'algorithmes). Notons qu'il y a deux raisons possibles de modéliser un système dynamique : dans le but de le contrôler (cf. chapitre II.9 sur la planification) ou dans le but de l'analyser, comme c'est le cas en diagnostic ou en supervision (cf. chapitre 15 sur le diagnostic). Selon le cas, les notions intéressantes ne sont pas les mêmes, cette différence détermine le choix du modèle. Par ailleurs, un même modèle de concept peut être associé à plusieurs représentations différentes. Si le choix d'un modèle dépend typiquement de l'information disponible et de ce que l'on veut en faire, celui de la représentation (pour un modèle fixé) s'appuie sur d'autres critères (efficacité calculatoire et efficacité spatiale, entre autres).

12.2 Raisonnement sur l'action : modèles

12.2.1 Les concepts de base et leurs modélisations

Dans cette partie, on définit les objets mathématiques correspondant aux concepts-clés pour le raisonnement sur l'action et le changement.

Le *modèle* d'un processus de raisonnement sur *un système dynamique* est constitué du modèle du système (avec sa dynamique) et du modèle de l'agent (avec ses connaissances). Sandewall [60] a élaboré une taxonomie des problèmes de raisonnement sur des systèmes dynamiques, dont la suite de ce chapitre s'inspire partiellement.

Dans tout le chapitre, on suppose que le temps est modélisé par une échelle discrète (ce qui est l'hypothèse la plus courante en intelligence artificielle). L'*horizon* du processus est l'ensemble $\mathcal{H}$ des étapes pertinentes pour le contrôle et l'observation du processus. Il peut être fini ($\mathcal{H} = \{0, \ldots, N\}$) ou infini ($\mathcal{H} = \mathbb{N}$) ; un cas dégénéré d'horizon fini est celui de l'horizon où il n'existe qu'une seule étape de changement ($\mathcal{H} = \{0, 1\}$).

Un *état* est la description du système à un instant donné. Sauf indication contraire, **l'ensemble des états, noté $\mathcal{S}$**, sera supposé fini. Une *trajectoire d'états* est une suite d'éléments de $\mathcal{S}$, indicés par des éléments de $\mathcal{H}$. Les états du système aux différents instants de $\mathcal{H}$ peuvent être plus ou moins bien connus par l'agent considéré.

La formalisation d'un problème de raisonnement sur un système dynamique inclut la modélisation des *croyances* que l'agent a sur l'état du système à différents instants (en particulier à l'instant initial), ainsi que sur des lois générales qui gouvernent son évolution. Il faut donc commencer par définir un modèle de l'incertitude, afin de définir formellement l'ensemble des *états de croyance* possibles de l'agent (appelés aussi états épistémiques). Faute de place, nous nous en tiendrons aux deux modèles d'incertitude suivants : **le modèle *binaire***, où les **états de croyance b** sont des *sous-ensembles non vides de* $\mathcal{S}$ et le **modèle bayésien**, où les états de croyance b sont des *distributions de probabilité* sur $\mathcal{S}$ [1].

Les transitions d'un état à un autre sont déclenchées par des *événements*. Ces événements font évoluer non seulement l'état du système, mais en général aussi les croyances de l'agent. Une *action* est un événement particulier qui est déclenché par un agent. L'agent possède un modèle de chacune des *actions* qui lui sont disponibles. L'ensemble des actions disponibles pour l'agent est noté $\mathcal{A}$, et, sauf indication contraire, est supposé fini. L'agent peut également disposer d'un modèle des

1. Il y a bien d'autres modèles de l'incertitude qui mériteraient d'être évoqués mais qui ne le seront pas, pour des raisons d'espace – notamment les modèles ordinaux, où les états de croyance et les effets des actions sont des relations de pré-ordre sur $\mathcal{S}$, les modèles possibilistes qui en sont très proches, les modèles probabilistes non bayésiens, où un état de croyance est une famille de distributions de probabilité, etc. (cf. chapitre 3).

événements exogènes, qui sont des phénomènes dont la dynamique est similaire à celle des actions mais qui s'en distinguent par le fait qu'ils ne sont pas déclenchés par l'agent, mais par la nature (ou par d'autres agents, plus ou moins bien identifiés et au sujet desquels l'agent de référence peut disposer d'informations imparfaites), et que leurs occurrences ne sont pas connues *a priori* de l'agent. On distingue le *type d'action* α (défini en toute généralité) de ses occurrences en un ou plusieurs instant(s) : une action donnée peut ne pas avoir d'occurrence dans une instance d'un problème, ou peut en avoir plusieurs. Les actions ont deux sortes d'effets : les effets *ontiques* (ou physiques), portant sur le monde, et les effets *épistémiques*, portant sur les croyances de l'agent. Ces derniers peuvent être provoqués par la connaissance que l'agent a *a priori* des effets physiques de l'action (je sais que l'action « détruire le fichier F » a pour effet que le fichier F ne figure plus sur mon ordinateur ; donc, lorsque j'exécute cette action, l'état de croyance résultant est tel que je sais que F n'est plus sur mon ordinateur). Les effets épistémiques peuvent aussi provenir d'*observations*, ou de toute forme de *feedback*[2] (si j'observe, après avoir tenté d'allumer la lumière en basculant l'interrupteur, que celle-ci ne s'allume pas, alors, dans mon nouvel état de croyance, je sais que l'ampoule est cassée ou qu'il n'y a pas de courant).

Une action a en général les deux types d'effets à la fois (comme dans le cas de l'action « basculer l'interrupteur » ci-dessus). Certaines actions, dites *purement épistémiques*, n'ont que des effets épistémiques, et aucun effet sur l'état du monde : par exemple, mesurer une température, ou interroger une base de données. D'autres actions, dites *purement ontiques*, ont certes des effets épistémiques (il est difficile d'imaginer des actions n'ayant pas d'effet épistémique, hormis l'action « ne rien faire »), mais ces effets épistémiques sont la simple projection, par l'agent, de ce qu'il connaît des effets ontiques de l'action (comme pour l'action « supprimer le fichier F » ci-dessus). En d'autres termes, une action purement ontique ne donne aucun *feedback* à l'agent : son état de croyance après l'exécution de l'action coïncide avec l'état de croyance qu'il pouvait prévoir avant d'exécuter l'action (« *what you foresee is what you get* »). Il est clair que toute action peut être décomposée de façon unique en une action purement ontique et une action purement épistémique. Sans perte de généralité, on peut donc faire l'hypothèse que chaque action disponible est soit purement ontique soit purement épistémique (et c'est ce que nous ferons dans le reste du chapitre, sauf mention contraire explicite).

Commençons par décrire les *actions purement ontiques*. Les effets d'une action purement ontique α sont définis par un *système de transition* entre états du monde, modélisé comme une **relation binaire R_α sur $\mathcal{S}$**.

Le cas le plus simple est celui des actions *déterministes et toujours exécutables* : le système de transition d'une action α est alors une application R_α de $\mathcal{S}$ dans $\mathcal{S}$. Celles qui sont *à effets conditionnels* sont celles qui ne sont pas constantes : l'état résultant après exécution de l'action dépend de l'état avant son exécution. Par exemple, l'action « éteindre la lampe » peut être considérée comme déterministe et inconditionnelle (si l'on considère qu'elle a toujours pour effet que la lampe soit éteinte après). « Basculer l'interrupteur » peut être considérée comme déterministe à effets conditionnels puisque selon l'état (allumé ou éteint) de la lampe avant l'action, il est inversé après (dans une modélisation où l'on considérerait qu'aucune panne ne peut se produire).

Plus généralement, les actions peuvent ne pas être toujours exécutables, ce qui signifie qu'il existe des états s tels que $R_\alpha(s) = \emptyset$, et/ou *non déterministes*, ce qui signifie qu'il existe des états s tels que $R_\alpha(s)$ contient plus d'un élément. On peut par exemple considérer que l'action « détruire le fichier F » n'est pas exécutable si le fichier n'existe pas, dans ce cas, le modélisateur définira l'effet de l'action seulement pour les états où le fichier existe et il sera interdit d'effectuer l'action dans les autres cas. Une autre modélisation serait à effets conditionnels où l'action serait l'identité dans les situations où le fichier F n'existe pas et conduirait à des états où le fichier est détruit sinon.

Dans le cas non déterministe, le modèle de transition choisi dépend de la nature de l'incertitude dont on veut rendre compte, et à chaque état initial correspond un état de croyance sur les états ultérieurs. Notons que le choix d'une modélisation déterministe ou non dépend des connaissances

2. Dans ce chapitre mouvementé, il y a sans doute assez d'actions pour que l'on s'autorise cet anglicisme afin d'éviter d'employer « réaction » ou « rétroaction ».

ou des objectifs du modélisateur, ainsi l'action « éteindre l'ordinateur » peut être considérée comme non déterministe pour un non-informaticien (puisqu'il arrive aléatoirement qu'après l'extinction l'ordinateur reste allumé) mais déterministe à effets conditionnels pour un expert en informatique (puisque cet expert saura dire dans quels cas après extinction, l'ordinateur demeure allumé). La modélisation par système de transition entre états fait l'hypothèse implicite que le système considéré est markovien [3], hypothèse qui est sans perte de généralité car tout système peut être modélisé de façon markovienne, en considérant des états du système plus complexes (codant des trajectoires d'états). Dans un souci de concision, nous nous en tiendrons aux deux types suivants : le modèle non déterministe binaire et le modèle stochastique.

Dans le modèle non déterministe binaire, le système de transition d'une action α est une application R_α de $\mathcal{S}$ dans $2^\mathcal{S}$ (ou dans $2^\mathcal{S}\backslash\{\emptyset\}$, si α est toujours exécutable). Par exemple, si les états du système sont $\mathcal{S} = \{o\_act, o\_veille, o\_eteint\}$ (représentant l'ordinateur en activité, en veille ou éteint) alors l'action d'éteindre pourrait se modéliser ainsi $R_{eteindre}(o\_act) = \{o\_act, o\_eteint\}$, $R_{eteindre}(o\_veille) = R_{eteindre}(o\_eteint) = \emptyset$ (signifiant qu'on ne peut éteindre l'ordinateur que s'il est en activité et que dans ce cas, on n'est même pas sûr de réussir à l'éteindre). Notons que si α est une action purement épistémique alors $R_\alpha(s) = \{s\}$ pour tout s.

Dans le modèle stochastique (correspondant au modèle de croyance bayésien), R_α est une matrice stochastique, c'est-à-dire une famille de distributions de probabilité $p(.|s, \alpha)$, pour $s \in \mathcal{S}$ où $p(s'|s, \alpha)$ est la probabilité qu'on obtienne l'état s' après avoir effectué α dans l'état s. Dans ce modèle, il est possible de préciser qu'éteindre donne plus souvent le résultat escompté, ainsi $R_{eteindre}$ pourrait être représenté par $p(o\_act|o\_act, eteindre) = 0.1$, $p(o\_eteint|o\_act, eteindre) = 0.9$, $p(o\_veille|o\_act, eteindre) = 0$.

Les effets *épistémiques* des actions s'expriment en termes de *feedback*. Les actions que l'agent décide d'exécuter sont fonction, non pas directement de l'état du système (qui peut être inconnu de l'agent) mais des croyances de l'agent (et en particulier, de ce qu'il en aura observé auparavant). Idéalement, l'état et ce qu'on en observe coïncident et les états de croyance de l'agent sont alors parfaits, mais cette hypothèse reflète un cas idéal qui est loin d'être la norme. Afin de définir les effets épistémiques des actions, on introduit dans la modélisation un **espace des observations Ω**, qui, sauf indication contraire, est supposé fini. Les observations représentent le *feedback* donné par le système et chaque observation effectuée à un instant donné de l'horizon constitue une projection d'un état (non nécessairement totalement observé) du système. Les observations sont dites *fiables* si cette projection correspond à l'*état réel* du système (les observations erronées peuvent provenir de capteurs défectueux, par exemple).

La prise en compte d'observations intervient lors de deux phases bien distinctes : la phase « hors ligne » de construction de la politique décisionnelle et la phase « en ligne » de mise en œuvre de la politique (l'exécution du « plan » proprement dite). Durant la phase hors ligne, l'agent de référence qui conçoit cette politique décisionnelle exploite la connaissance qu'il a des observations qui pourront être réalisées à l'exécution. Durant la phase « en ligne », les actions déclenchées dépendent typiquement des observations qui sont effectivement réalisées. L'agent de référence planifie mais n'exécute pas nécessairement lui-même les plans qu'il conçoit.

Deux hypothèses extrêmes sont courantes : dans le cas d'un système *totalement observable* l'espace des observations est identique à celui des états : quand il planifie, l'agent de référence sait qu'à l'exécution l'agent qui s'en chargera connaîtra à tout instant l'état courant du système avec précision ; dans le cas d'un système *non observable*, l'espace des observations est un singleton $\{o^*\}$, où o^* est une observation fictive (observation vide) : le système ne donnera aucun *feedback*.

En l'absence de l'une de ces deux hypothèses extrêmes, on est dans la situation plus générale d'observabilité partielle, où les observations et les états sont liés par une *structure de corrélation* observation-état, dont le modèle varie en fonction du modèle d'incertitude. De façon générale, à chaque état s et chaque *action épistémique* α correspond un **état de croyance $O_\alpha(s)$** sur l'espace des observations, représentant les croyances a priori sur l'observation obtenue lorsque l'action α est effectuée dans l'état s. Dans le modèle binaire, on a donc une famille d'ensembles $O_\alpha(s)$, pour

3. Un système est dit « markovien » si la transition vers un état du système ne dépend que de l'état courant indépendamment des états antérieurs.

$s \in \mathcal{S}$, où $O_\alpha(s)$ *est un sous-ensemble non vide de* Ω ; $o \in O_\alpha(s)$ traduit le fait que l'état s est un état compatible avec l'observation o réalisée grâce à l'exécution de α. Si α est purement ontique alors $O_\alpha(s) = \{o^*\}$ pour tout s. Dans le modèle d'incertitude bayésien, le *feedback* est modélisé par une distribution de probabilité $p(.|s, \alpha)$ sur Ω où $\boldsymbol{p}(o|s, \alpha)$ *est la probabilité d'observer* o *quand on exécute l'action* α *dans l'état* s.

Dans cette section, on a fait l'hypothèse qu'on pouvait exécuter seulement une action à la fois. Dans certains problèmes, il est naturel de pouvoir exécuter plusieurs actions de façon *concurrente*. Cela nécessite d'être à même de définir les effets des combinaisons d'actions ; pour cela, on peut utiliser les mêmes modèles que précédemment, en considérant chaque combinaison possible d'actions comme une action différente. Un exemple typique [67] est celui d'une table que l'on peut soulever par le côté droit ou par le côté gauche, les deux actions effectuées en séquence n'ayant pas le même effet que si elles le sont simultanément, en particulier lorsqu'un verre d'eau est posé sur la table.

12.2.2 Les types de raisonnement et leurs mises en œuvre

Raisonner sur un système dynamique nécessite la prise en compte d'un horizon, de croyances *a priori* sur le système (les lois générales, la dynamique : les effets des actions, ...), d'occurrences d'actions en certains instants, et d'observations en certains instants (c'est un modèle simplifié – voir [60] pour un modèle plus général, où, notamment, les actions peuvent avoir une durée). Nous abordons maintenant quelques types de raisonnement spécifiques impliquant le raisonnement sur un système dynamique ainsi que leur mise en œuvre au moyen d'algorithmes.

Prédiction et postdiction à partir d'actions ontiques

La *prédiction* (appelée aussi *projection*) consiste à déterminer, en fonction d'un état de croyance initial b et de la description d'une action purement ontique α, le nouvel état de croyance b' résultant de l'application de α dans b. La transformation d'un état de croyance en un autre par une action est appelée *progression* ; on notera $b' = prog(b, \alpha)$. Bien entendu, la définition formelle de *prog* dépend de la nature de l'espace des croyances (statiques et dynamiques), donc du modèle d'incertitude choisi. Dans le cas le plus simple (celui de la planification classique) où les états de croyance sont parfaits, les actions déterministes et toujours exécutables, chaque $prog(., \alpha)$ est une application associant un état à un autre. Dans le modèle non déterministe binaire, un état s' est possible après l'exécution de α dans l'état de croyance $b \subseteq \mathcal{S}$ s'il existe un état possible s dans l'ensemble d'états correspondant à la croyance initiale b, tel que s' est un résultat possible de α dans s, c'est-à-dire $prog(b, \alpha) = \bigcup_{s \in b} R_\alpha(s)$. Dans le modèle probabiliste, le choix évident est obtenu en identifiant le modèle du processus à une chaîne de Markov : $prog(b, \alpha)$ est la distribution de probabilité b' sur $\mathcal{S}$ définie par $b'(s') = \sum_{s \in \mathcal{S}} b(s)p(s'|s, \alpha)$ (où $p(.|s, \alpha)$ est la distribution de probabilité associée à R_α).

Le deuxième type de raisonnement est la *postdiction*. Elle consiste à déterminer, en fonction d'un état de croyance final b' et de la description d'une action purement ontique α qui vient d'être effectuée, l'état de croyance b avant que l'action ait été effectuée. Cette transformation d'un état de croyance en un autre, s'appelle parfois aussi *régression* ou *régression faible* ; on notera $b = reg_f(b', \alpha)$. La régression faible correspond à la progression par l'action inverse de α (notée α^{-1}), dont le système de transition $R_{\alpha^{-1}}$ est la relation réciproque de la relation R_α ; on a donc $reg_f(b', \alpha) = prog(b', \alpha^{-1}) = \{s | R_\alpha(s) \cap b' \neq \emptyset\}$.

La postdiction doit être distinguée de la *régression de but*, appelée aussi *régression forte*, qui est la transformation inverse de la progression. Elle n'est définie que dans le modèle binaire [4] : étant donné un état de croyance $b' \subseteq \mathcal{S}$ et une action purement ontique α, il s'agit de déterminer l'état de croyance $b = reg_F(b', \alpha)$ tel que $prog(b, \alpha) \subseteq b'$ et b est maximal pour l'inclusion ensembliste ; cet état de croyance est l'état de croyance le moins informatif (donc le moins conjectural) qui garantit que l'exécution de α dans celui-ci conduit au but b'.

4. Dans le modèle probabiliste, il n'existe en effet pas forcément une distribution de probabilité unique b sur $\mathcal{S}$ vérifiant $b'(s') = \sum_{s \in \mathcal{S}} b(s)p(s'|s, \alpha)$, $b'(s')$ et $p(s'|s, \alpha)$ étant connues pour tous s, s' et α.

On remarquera que $reg_F(b', \alpha) \subseteq reg_f(b', \alpha)$ et que $reg_F(b', \alpha) = reg_f(b', \alpha)$ lorsque α est déterministe.

Progression et régression forte sont deux processus de raisonnement clés pour la *planification* d'actions (cf. chapitre II.9), qui consiste à déterminer les actions à exécuter pour faire évoluer le système comme l'agent le souhaite (par exemple, « coller au mieux » à une trajectoire de référence dans le cas de la supervision, ou atteindre un état but dans le cas de la planification classique). En revanche, la postdiction est de peu d'intérêt pour la planification elle-même (car si b est obtenu comme possible postdiction depuis b' avec l'action α, il n'est pas garanti qu'en effectuant l'action α on ré-obtienne forcément l'état b', alors que la régression forte le garantit (par définition)).

Prédiction et postdiction à partir d'actions épistémiques

La progression d'un état de croyance par une action épistémique dépend de la nature du processus impliqué. Dans le cas d'un processus de supervision ou de diagnostic, l'agent raisonne en ligne et dispose donc de toutes les observations provenant du *feedback* des actions au moment où il raisonne ; il suffit donc de définir la progression d'un état de croyance par une observation, ce qui s'apparente à une *révision des croyances* (cf. chapitre 11). Dans le modèle binaire, la progression d'un état de croyance $b \subseteq \mathcal{S}$ par une observation o après avoir effectué l'action α est $b \cap S(o)$, où $S(o) = \{s | o \in O_\alpha(s)\}$; tandis que dans le modèle bayésien, la révision de b par o est la distribution de probabilité $b(.|S(o))$.

Le processus de *filtrage* consiste à déterminer le nouvel état de croyance b', étant donné un état de croyance initial b, une action α, et une observation o résultant de l'exécution de α. Dans le modèle binaire, ce nouvel état de croyance est tout simplement $prog(b, \alpha) \cap S(o)$. Dans le modèle bayésien, la distribution de probabilité b' obtenue après avoir effectué α et observé o est $b'(s') = \frac{p(o|s',\alpha).\sum_{s \in S} b(s).p(s'|s,\alpha)}{\sum_{s'' \in S}(p(o|s'',\alpha).\sum_{s \in S} b(s).p(s''|s,\alpha))}$; c'est la formule permettant la révision des croyances par le *feedback* dans les processus décisionnels de Markov partiellement observables (cf. chapitre II.9).

Dans le cas d'un processus de planification, où il s'agit de construire un plan hors ligne et de raisonner sur ses effets, la progression d'un état de croyance par une action épistémique n'est en général pas un état de croyance unique, mais un ensemble de tels états (un pour chaque observation possible, puisque l'observation effective ne peut pas être connue hors ligne). Dans le modèle binaire, $prog(b, \alpha)$ est l'ensemble des états de croyance $\{b \cap S(o) \mid b \cap S(o) \neq \emptyset\}$ pour o variant dans Ω. Par manque de place, nous ne donnons pas de détails sur la régression par des actions épistémiques.

Abduction d'événements

Le troisième type de raisonnement est l'*abduction d'événements*. Il s'agit de raisonner sur l'événement qui a eu lieu entre deux instants successifs t et $t+1$ de l'horizon, à partir de la description des événements possibles et des états de croyance à l'instant t et à l'instant $t+1$ qui le suit [5]. Si l'événement en question est exogène, on parle d'*explication*.

Comme pour la planification, progression et régression de but sont deux processus clés pour l'abduction d'événements : quand on planifie, on doit choisir les actions à réaliser pour faire évoluer le système comme souhaité depuis son état courant ; en abduction d'événements, l'objectif est de déterminer quel événement α a conduit le système à évoluer comme il a évolué entre t et $t+1$ (même si cette évolution n'était pas du tout souhaitée). Dans le modèle binaire, calculer un tel α consiste à rechercher parmi les événements possibles ceux qui vérifient $b' \subseteq prog(b, \alpha)$ (ou encore $b \supseteq reg_F(b', \alpha)$).

Extrapolation, suivi de scénario

Plus généralement, ces types de raisonnement, que nous avons définis dans le contexte où il n'y a qu'une seule étape de changement (donc deux instants), ont lieu dans des situations

5. Un problème plus complexe d'abduction consiste à raisonner non seulement sur l'événement qui a eu lieu, mais aussi sur les états du système aux instants t et $t+1$, sur lesquels on désire obtenir des croyances plus précises.

où l'horizon est quelconque, et où l'information d'entrée est un *scénario* complexe consistant en une trajectoire partielle du système (la donnée éventuelle, pour chaque instant, d'une occurrence d'action et/ou d'une observation). Dans le cas particulier où aucune action n'a été effectuée et où l'on cherche les événements (ou plus simplement, les changements élémentaires) qui se sont produits en chaque instant, le processus s'appelle *extrapolation*. Une autre situation est celle où l'on cherche à reconnaître des trajectoires parmi un ensemble de trajectoires de référence afin de prédire les événements qui se produiront ensuite et/ou les états vers lesquels le système va évoluer ; ce processus s'appelle *suivi* ou *reconnaissance de scénarios*.

Un aspect crucial des approches du raisonnement sur le changement en intelligence artificielle est qu'elles donnent une place importante à l'*inertie* : par défaut, le système a tendance à rester statique, et les changements autres que ceux qui sont directement causés par les occurrences d'action sont rares, c'est pourquoi on cherche à les minimiser. Cette hypothèse est cruciale si l'on veut effectivement raisonner sur l'action en présence d'incertitude sans perdre trop d'information. Bien souvent, raisonner sur le changement revient avant tout à *minimiser les changements* ; nous y reviendrons lorsque nous aborderons les langages de raisonnement sur l'action. En effet, selon la représentation choisie pour les actions, il existe de nombreuses façons de réaliser la progression d'un état ou la régression d'une formule, codant un ensemble d'états, par une action. Pour les représentations en logique temporelle ou dynamique, progression et régression peuvent être calculées via quelques transformations de formules (en particulier, la conjonction et l'oubli). L'utilisation de principes de minimisation des changements est souvent proposée comme un moyen de résoudre le problème du décor (cf. section 12.3.1), mais il semble dorénavant admis qu'il faut plutôt mettre en place des procédés qui suppriment les solutions contenant des changements anormaux (non provoqués par des actions) plutôt que des procédés qui les minimisent.

12.3 Raisonnement sur l'action : langages

12.3.1 Problèmes liés à la représentation des actions

Dans la majorité des problèmes réels, le système est naturellement décrit par un certain nombre de variables, dites variables d'état, représentant des objets, des propriétés, etc. Dans ce cas, un état du système consiste en la donnée d'une valeur pour chacune de ces variables, ces valeurs pouvant changer au cours du temps. On appelle généralement *fluents* ces variables décrivant une propriété dynamique du système. Bien évidemment, le nombre d'états possibles est exponentiel en le nombre de variables. La description *explicite* des effets des actions, qui consiste à spécifier *in extenso* les fonctions R_α, devient alors infaisable en pratique, et est de surcroît assez peu naturelle, car elle force à décrire les actions état par état. Des considérations similaires s'appliquent à la description des corrélations entre états et observations et au calcul des opérations de progression, de régression, etc.

Or, souvent, il existe des façons bien plus économiques et naturelles de représenter les effets des actions. Par exemple, considérons l'action qui consiste à changer basculer un interrupteur provoquant alternativement l'allumage ou l'extinction d'une ampoule. Si la prise en compte de l'état allumé-éteint de 10 ampoules est nécessaire à la représentation du problème, il faudra considérer 1024 états du système (toutes les configurations possibles des 10 ampoules) pour décrire l'action, alors que celle-ci ne provoque le changement d'état que d'une ampoule particulière. Pour décrire une telle action, on a plutôt envie de se limiter à indiquer qu'elle change l'état de cette ampoule et, implicitement, qu'elle laisse les autres ampoules dans leur état.

Les *langages d'action* ont été construits précisément dans cet objectif : permettre une représentation à la fois plus économique (ou plus compacte) et plus naturelle des effets des actions. Le problème de permettre d'éviter de décrire explicitement les fluents qu'une action laisse inchangée dans les différents contextes possibles est connu sous le nom du *problème du décor* (ou *frame problem* [53]). Il s'agit bien d'un problème lié au choix d'une représentation des actions (et non pas d'un problème de modélisation, c'est-à-dire que le problème ne vient pas du choix des fluents utilisés pour modéliser le système mais du codage des actions en général).

Dans le même registre, on trouve aussi le problème dual du problème du décor, dit *problème de la ramification* [21] qui est résolu lorsque le langage d'actions utilisé permet d'éviter de décrire explicitement les fluents qu'une action modifie dans les différents contextes possibles. Pour poursuivre l'exemple précédent, chaque fois que basculer l'interrupteur provoque l'allumage de l'ampoule associée, alors la pièce où l'ampoule se situe devient éclairée et par conséquent on peut s'y installer pour lire. Ce fait dérivé est une conséquence de l'exécution de l'action mais il n'est pas naturel, quand on décrit l'action, de le spécifier directement : il résulte plutôt d'une loi (statique) qui fait que lorsqu'une pièce est éclairée, on peut y pratiquer la lecture.

Quand on traite de représentation d'actions, on évoque souvent aussi le problème dit de la *qualification* [51] ; ce problème exprime l'incapacité à décrire toutes les pré-conditions nécessaires pour garantir l'obtention de l'effet « normal » d'une action. Pour traiter ce problème, il faut d'abord circonscrire le monde aux individus et aux propriétés explicitement présents dans la représentation ; par exemple, basculer l'interrupteur lorsque l'ampoule associée est éteinte ne provoquera l'allumage de celle-ci que si le conflit entre Bordures et Syldaves n'a pas provoqué la destruction de la ligne électrique alimentant la maison. De notre point de vue, ce problème n'est pas intrinsèque à la représentation d'actions, il se pose plus primitivement dès la phase de modélisation et reflète simplement la différence existant entre une situation du monde physique et une modélisation de celle-ci, qui l'abstrait nécessairement. Cependant, pour donner les pré-conditions d'une action, cette restriction aux situations représentables dans le langage ne supprime pas la nécessité de passer en revue toutes les situations dans lesquelles l'action s'effectue normalement. Résoudre le problème de la qualification des actions, c'est pouvoir énoncer les pré-conditions « naturelles » d'une action sans avoir à décrire explicitement la liste de toutes les valeurs des fluents qui permettent que l'action se déroule normalement.

Une fois les actions représentées, il faut construire des algorithmes permettant le calcul des opérations de base (progression, régression, etc.). Le choix d'un langage d'actions dépend donc, d'une part, de son aspect plus ou moins naturel, d'autre part, de sa compacité (ou efficacité spatiale), et enfin, de la complexité des opérations de base lorsque les actions sont représentées dans ce langage (son efficacité calculatoire).

Il existe de nombreux langages d'actions qui ont été développés et étudiés par la communauté. Ils peuvent être regroupés en plusieurs familles, selon la nature des objets mathématiques qu'ils utilisent (formules logiques propositionnelles ou du premier ordre, formules de la logique temporelle ou dynamique, réseaux bayésiens, automates, etc.). En donner un panorama exhaustif serait trop long et peu digeste. Nous nous contenterons donc de présenter sommairement les langages qui ont reçu le plus d'attention de la part de la communauté, et qui sont suffisamment représentatifs de l'éventail des langages existants. Chacune des sous-sections suivantes aborde un langage particulier, ou une famille de langages, en en présentant succinctement les spécificités.

12.3.2 Le calcul des situations

Historiquement le *calcul des situations* introduit par McCarthy et Hayes [53] est le premier formalisme dédié au raisonnement sur les actions. Les définitions proposées par ces auteurs ont permis de fixer les concepts de base (présentés plus haut) du raisonnement sur le changement et l'action. Le calcul des situations est un langage typé de la logique du premier ordre avec égalité, dont les types sont les fluents, les états (appelés situations), les actions et les objets. Afin de simplifier la présentation, nous considérons ici seulement des fluents propositionnels, ayant comme unique argument une situation ; nous n'avons donc pas besoin de noms d'objets. Ainsi, $\neg P(S_0)$ exprime que le fluent P est faux dans la situation S_0. S_0 nomme l'état du système à l'instant initial de l'horizon. Pour les situations et les actions nous avons besoin à la fois de variables (notées respectivement $s, \ldots$ et $x, \ldots$) et de constantes (notées respectivement $S, \ldots$ et $A, \ldots$). La fonction *do* s'applique à une situation et une action et renvoie une situation. Ainsi, la formule $\neg P(S_0) \wedge P(do(A_1, S_0))$ exprime que P est faux en S_0 et vrai en $do(A_1, S_0)$, c'est-à-dire dans la situation obtenue en appliquant l'action A_1 en S_0. La formule $\forall s \neg P(s)$ exprime que P est toujours faux. La formule $\forall s((\forall x \neg P(do(x, s)) \leftrightarrow x = A_0)$ exprime que A_0 est l'unique action qui garantit de rendre P faux dans tout état où on l'applique.

McCarthy et Hayes ont posé un cadre général de représentation permettant de représenter des actions par leurs pré-conditions et leurs effets (représentés par des formules logiques). De multiples approches ont ensuite été proposées afin de caractériser les « bonnes » conséquences de ces formules. Dans un premier temps, tous les auteurs misaient sur la *minimisation des changements* pour contraindre l'ensemble des modèles afin que les propriétés résultant du principe d'inertie puissent être déduites sans être explicitées. Ceci était accompli à l'aide d'une formule du second ordre, et différentes politiques de circonscription ont été étudiées à cette fin (se reporter à [54] pour un résumé). McCarthy [52] puis Hanks et McDermott [31] utilisèrent la circonscription des prédicats d'anormalité (en considérant qu'un fluent doit persister sauf mention explicite du contraire) dans le cadre du calcul des situations. Cependant, la circonscription ne donne pas le résultat attendu sur certains exemples. Un des plus fameux est le *Yale Shooting Problem* proposé par Hanks et McDermott : une personne donnée est vivante dans la situation initiale, et on effectue successivement les trois actions « Charger », « Attendre » puis « Tirer ». L'action « Tirer » est décrite par la formule : $\forall s$, (Chargé(s) $\rightarrow$ (Anormal(Vivant, Tirer, s) $\wedge \neg$ Vivant(do(Tirer, s)))) [6]. Le fait que, par défaut, les fluents persistent est décrit par la formule du second ordre $\forall f, s, a$, ((f(s) $\wedge \neg$ Anormal(f,a,s) $\rightarrow$ f(do(a,s)) [7]. La circonscription du prédicat *Anormal* permet d'obtenir un modèle logique dans lequel la personne est vivante à l'instant initial et morte (non vivante) après l'action « Tirer ». Cependant, un autre modèle est possible : celui où le fusil s'est déchargé pendant l'attente et la personne est encore vivante après « Tirer ». La circonscription du prédicat d'anormalité ne permet pas de préférer le premier modèle au second car les deux modèles ont des ensembles d'anormalités incomparables pour l'inclusion ensembliste (dans le premier, c'est « Vivant » qui est anormal par rapport à « Tirer » ; dans le second c'est « Chargé » qui est anormal par rapport à « Attendre »). L'ignorance chronologique proposée par Shoham [62] et consistant à autoriser les changements le plus tard possible permet d'obtenir une réponse satisfaisante à ce problème. Mais cette approche *ad hoc* fut ensuite remise en cause par d'autres exemples qu'elle traite mal [60, 23].

Une autre solution proposée par Lifschitz et Rabinov [47] est d'imposer que tous les fluents modifiés par une action soient systématiquement non inertes quand cette action est effectuée. Cette idée est proche de la solution proposée par Castillo, Gasquet et Herzig [11] d'utiliser une relation de dépendance entre une action et les atomes sur lesquels elle peut agir. Le lecteur peut se reporter à [60] pour une excellente synthèse de tous ces travaux.

En résumé, les approches issues de la minimisation des changements s'appuient sur des logiques *non monotones* et fort complexes ; elles ne permettent pas de déduire toutes les conséquences intuitives d'une description d'actions et d'une situation initiale.

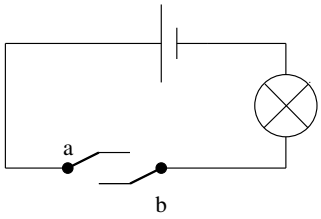
La situation a changé avec la publication de ce qui a été appelé la *solution de Reiter* au problème du décor [59]. Reiter propose une solution *monotone* à base d'axiomes de l'état suivant (*successor state axioms*, SSA). Ces axiomes doivent être donnés pour chaque fluent P (ce qui équivaut à une hypothèse d'information complète sur les conditions de changement de valeur de vérité d'un fluent) et prennent la forme

$$\forall s, x \quad (P(do(x, s)) \leftrightarrow \gamma_P(x, s))$$

où $\gamma_P(x, s)$ est une formule ne contenant pas la fonction *do* et ne pouvant contenir que S_0 comme seule constante de situation. Ainsi, le SSA pour P décrit les conditions sous lesquelles P est vrai après une action, en termes de ce qui était vrai avant.

Considérons l'exemple du va-et-vient [46] :

Dans une salle donnée, la lumière est allumée seulement si les deux interrupteurs sont tous les deux en haut ou tous les deux en bas. Initialement, l'interrupteur a est en haut et l'interrupteur b en bas, la lampe est donc éteinte, on bascule l'interrupteur a.



6. Si le fusil est chargé dans la situation s alors le fluent « Vivant » est anormal (i.e., non persistant) lorsque l'action « Tirer » a lieu en s et la personne ne sera plus vivante dans la situation résultante.

7. Si un fluent n'est pas anormal vis-à-vis d'une action alors il garde sa valeur après exécution de cette action.

Les fluents sont U_a (« l'interrupteur a est en haut ») et U_b (« l'interrupteur b est en haut »). Dans cet exemple, le SSA pour le fluent U_a peut s'écrire :

$$\forall s, x \quad U_a(do(x, s)) \leftrightarrow ((\neg U_a(s) \wedge x = B_a) \vee (U_a(s) \wedge x \neq B_a))$$

où B_a est l'action de basculer l'interrupteur a, i.e., le changer de position. Reiter explique que de telles SSA constituent une solution au problème du décor car on peut raisonnablement s'attendre à ce que la taille de l'ensemble des SSA soit de l'ordre du cardinal de l'ensemble des fluents (ce qui contraste avec la taille de la description explicite des axiomes de décor qui est de l'ordre du produit du cardinal de l'ensemble des fluents avec le cardinal de l'ensemble des actions).

Selon Reiter, la quantification sur les actions est la clé de sa solution au problème du décor. Comme nous allons le montrer dans le paragraphe sur la logique dynamique, c'est plutôt l'hypothèse d'information complète sur les conditions de changement de valeur de vérité d'un fluent (traduite par les $\leftrightarrow$ dans les SSA) qui explique que la solution de Reiter traite le problème du décor de façon satisfaisante. La présence de SSA pour chaque fluent permet de *régresser* les formules : les atomes de la forme $P(do(\alpha, \sigma))$ (où α et σ sont des termes construits à partir de variables, de constantes et de la fonction do) sont remplacés par le membre droit du SSA pour P, en appliquant d'abord la substitution appropriée ; ce processus est itéré jusqu'à élimination complète de la fonction do. Par construction, la formule ainsi obtenue concerne seulement l'état initial S_0. Par exemple, la formule

$$U_a(do(B_a, do(B_a, S_0)))$$

est, dans un premier temps, remplacée par :

$$(\neg U_a(do(B_a, S_0)) \wedge B_a{=}B_a) \vee (U_a(do(B_a, S_0)) \wedge B_a{\neq}B_a)$$

qui peut être simplifiée en $\neg U_a(do(B_a, S_0))$. Dans un second temps, cette dernière formule est remplacée par $\neg(\neg U_a(S_0) \wedge B_a{=}B_a) \vee (U_a(S_0) \wedge B_a{\neq}B_a)$ qui peut être simplifiée en $U_a(S_0)$. Nous avons ainsi démontré par régression que l'interrupteur est en position haute après deux exécutions de B_a si et seulement s'il est en position haute dans l'état initial S_0.

Pour décider si l'application de l'action α dans l'état S_0 conduit à un état dans lequel ψ est satisfait, il suffit de décider si la formule $\phi(S_0) \rightarrow \psi(do(\alpha, S_0))$ est valide, La régression de $\psi(do(\alpha, S_0))$ résulte en une formule $\psi'(S_0)$. Si on élimine l'argument S_0, on obtient la formule propositionnelle $\phi \rightarrow \psi'$ dont on peut tester la validité en utilisant un démonstrateur approprié.

Cette solution a été combinée avec la logique épistémique [61], ce en quoi les auteurs se rapprochent de la logique dynamique, formalisme que nous abordons au paragraphe 12.3.4.

12.3.3 Langages d'actions propositionnels

Un point faible des approches fondées sur le calcul des situations est leur difficulté de mise en œuvre algorithmique. Pour cette raison, les chercheurs ont également développé des approches fondées sur la logique propositionnelle, permettant ainsi l'utilisation de prouveurs SAT ou ASP (cf. chapitres II.4et II.5).

Les langages d'action fondés sur la logique propositionnelle représentent les effets des actions par des règles locales spécifiant uniquement les fluents qui changent, avec éventuellement les conditions dans lesquelles ils changent. Soit F un ensemble fini de fluents. Les états de $\mathcal{S}$ sont les interprétations propositionnelles, c'est-à-dire que $\mathcal{S} = 2^F$.

Le langage d'action le plus rudimentaire est le formalisme *STRIPS* [20], dans lequel une action est représentée par ses pré-conditions et ses effets (cf. chapitre II.9sur la planification), une pré-condition étant une conjonction de littéraux et un effet étant un ensemble cohérent de littéraux.

Pour coder l'exemple du va-et-vient, on peut considérer que les littéraux sont $F = \{U_a, U_b\}$, U_a (resp. U_b) signifiant que l'interrupteur a (resp. b) est en haut. Une action à effets conditionnels comme B_a (« basculer a ») peut s'écrire sous la forme d'une conjonction du type $(U_A \mapsto \neg U_A) \wedge (\neg U_A \mapsto U_A)$. Le membre droit l de chaque conjoint de la forme $c \mapsto l$ constitue un effet direct de l'action, qui s'applique si et seulement si la condition correspondante est satisfaite dans l'état où l'action a lieu. Ainsi, appliquer $c \mapsto l$ dans un état s conduit à laisser s inchangé si s ne satisfait pas c et à forcer l dans s sinon, en laissant les autres fluents inchangés. Cela s'applique pour

chaque conjoint [8]. Ainsi, appliquer B_A dans un état s conduit à changer la valeur de vérité de U_A dans s, comme on s'y attend. Il est essentiel de comprendre qu'une telle description d'action n'est pas une formule de la logique classique et en particulier que $\mapsto$ n'est pas l'implication matérielle. En effet, une action STRIPS α peut être vue comme une contrainte liant l'état du monde *avant* exécution de l'action à l'état du monde *après* son exécution. Ainsi, contrairement à ce qui se passe pour l'implication matérielle, $c \mapsto l$ n'équivaut pas en général à ce qui serait sa « contraposée ». [9]

Une des limites du langage STRIPS est l'impossibilité d'exprimer des lois statiques. Ces lois sont pourtant essentielles pour traiter le problème de la ramification . Ainsi, dans l'exemple précédent, on peut vouloir s'intéresser à un nouveau fluent L représentant « la lampe est allumée ». Si l'on utilisait le langage STRIPS pour intégrer ce nouveau fluent cela nécessiterait de modifier toutes les actions en précisant ce qui se passe pour le fluent L. Cette solution n'est pas envisageable lorsqu'on dispose d'un grand nombre de fluents. Un palliatif à ce manque d'expressivité est de ne coder les actions qu'à partir d'un ensemble de fluents de base. Les fluents de base sont ceux sur lesquels les actions disponibles agissent directement (U_a et U_b dans l'exemple). STRIPS permet de calculer des progressions. Il faut ensuite rajouter une étape de calcul afin d'intégrer les lois statiques et pouvoir faire des déductions sur les fluents dérivés. Ainsi, pour calculer la progression d'un état s par une action, on commence par projeter s sur ses fluents de base. Puis on effectue la progression de s proprement dite, pour enfin compléter la description obtenue en utilisant les lois statiques. Dans l'exemple du va-et-vient, on peut considérer qu'on a comme loi statique $((U_a \wedge U_b) \vee (\neg U_a \wedge \neg U_b)) \leftrightarrow L$. La progression de l'état $\{U_a, \neg U_b, \neg L\}$ par l'action B_a est donc $\{\neg U_a, \neg U_b, L\}$.

Le problème principal de STRIPS est son manque d'expressivité. En particulier, STRIPS ne permet pas de représenter (a) le non-déterminisme, (b) les relations causales statiques entre fluents (comme on vient de le voir dans le paragraphe précédent), (c) les actions concurrentes et (d) les actions épistémiques. Pour pallier ces défauts d'expressivité, des langages d'actions plus sophistiqués ont été développés, tant dans la communauté de la planification (avec ADL [58] et PDDL [26]) que dans celle de la représentation des connaissances. Nous allons nous attarder un peu sur les langages issus de cette dernière famille.

Dans les années 70 à 90, dans le domaine de la représentation des connaissances, la représentation des actions s'est appuyée sur l'implication matérielle entre pré-conditions et effets directs de l'action. Ensuite, les chercheurs ont proposé de calculer les prédictions en utilisant une *minimisation des changements* afin d'imposer que, par défaut, les fluents non touchés par l'action persistent (ces fluents n'étant pas mentionnés dans les effets pour éviter le problème du décor). Enfin, depuis les années 90, la minimisation des changements a laissé place à l'utilisation de langages propositionnels fondés sur l'*implication causale*. Ainsi, les solutions de [59] et [47] et [11] pour résoudre les problèmes liés à la minimisation des changements reviennent en définitive à exprimer la dépendance entre l'action et ses effets. C'est ce principe qui est mis en œuvre dans les travaux utilisant l'*implication causale*, distincte de l'implication matérielle puisque destinée à coder cette dépendance.

Certaines approches utilisant l'implication causale se situent dans le cadre du calcul des situations [66], [48]. D'autres utilisent la modalité C [24, 27, 69] ou de manière équivalente définissent un nouveau connecteur $\Rightarrow$ [28]. Certaines définissent des relations d'influence entre fluents [68]. La principale caractéristique de ces approches est de distinguer le fait d'être vrai du fait d'avoir une cause pour l'être, afin d'utiliser cette distinction pour calculer l'effet attendu des actions pour la prédiction ou la planification.

Nous détaillons ici le langage d'actions $\mathcal{A}$ proposé par Gelfond et Lifschitz [25]. Ce langage décrit une action au moyen de *règles causales* conditionnelles de la forme si c alors α CAUSE l, où α est un nom d'action, c une conjonction de littéraux (que l'on omet lorsqu'elle est équivalente à

8. Un cas pathologique est celui où des conditions associées à des littéraux complémentaires sont conjointement satisfaites dans s ; dans un tel cas, la progression n'est pas définie, ce qui peut refléter une erreur dans la représentation de l'action, ou le fait que s est impossible (c'est une loi statique supplémentaire, implicite).

9. Si tel était le cas alors le codage de l'action « tirer » par *chargé* $\mapsto \neg$ *vivant* dans le *Yale Shooting Problem* contraposé en *vivant* $\mapsto \neg$ *chargé*, indiquerait que lorsque l'on tire sur une personne vivante alors c'est que le fusil est déchargé (et donc que la personne restera vivante (le fluent « Vivant » persisterait)).

la tautologie $\top$), et l un littéral. Un ensemble de règles causales définit un système de transition déterministe entre états. Ainsi, l'action α définie par les règles causales si $p \wedge q$ alors α CAUSE $\neg p$, si $\neg p \wedge q$ alors α CAUSE p, α CAUSE q correspond au système de transition R_α défini par $R_\alpha(pq) = R_\alpha(\bar{p}\bar{q}) = \bar{p}q$ et $R_\alpha(\bar{p}q) = R_\alpha(p\bar{q}) = pq$. Une action α décrite par de telles règles causales correspond à une *théorie d'action propositionnelle* Σ_α, qui exprime α au moyen des symboles propositionnels F_t et F_{t+1}, avec $F_t = \{f_t | f \in F\}$ et $F_{t+1} = \{f_{t+1} | f \in F\}$, où f_t représente le fluent f à l'instant t, c'est-à-dire avant que l'action α ait été effectuée, et f_{t+1} représente f à l'instant $t+1$, après que l'action α a été effectuée. La traduction des règles causales en Σ_α est accomplie selon le principe suivant : le fluent f est vrai à $t+1$ si et seulement si l'une de ces deux conditions est réalisée : (a) il était vrai en t et l'état en t ne satisfaisait aucune condition de règle causale dont la conclusion est $\neg f$, ou (b) il était faux en t et l'état en t satisfaisait la condition d'une règle causale dont la conclusion est f. On retrouve ici le principe de base mis en œuvre dans le calcul des situations, que nous avons appelé « solution de Reiter » en section 12.3.2.

Formellement, soit $\Gamma(f)$ (respectivement $\Gamma(\neg f)$) la disjonction de toutes les conditions des règles dont la conclusion est f (respectivement $\neg f$) ; alors Σ_α est la conjonction de toutes les formules $f_{t+1} \leftrightarrow \Gamma(f)_t \vee (f_t \wedge \neg\Gamma(\neg f)_t)$ pour $f \in F$. Ainsi, la théorie d'action Σ_α correspondant à l'action α précédemment décrite par ses règles causales est $\Sigma_\alpha = (p_{t+1} \leftrightarrow ((\neg p_t \wedge q_t) \vee (p_t \wedge \neg(p_t \wedge q_t)))) \wedge (q_{t+1} \leftrightarrow \top)$, ce qui se simplifie en $\Sigma_\alpha = q_{t+1} \wedge (p_{t+1} \leftrightarrow (p_t \leftrightarrow \neg q_t))$.

Les successeurs du langage $\mathcal{A}$, comme le langage $\mathcal{C}$ [29] par exemple, permettent d'exprimer des conditions d'exécutabilité et des règles statiques, indépendantes de toute action, comme dehors $\wedge$ $\neg$ parapluie $\wedge$ pluie CAUSE $\neg$ sec, qui sont également prises en compte dans la théorie d'action Σ_α. Par exemple, considérons l'action sortir ayant pour unique règle causale sortir CAUSE dehors ; la théorie d'action correspondante, tenant compte de la règle causale statique précédente, est Σ_{sortir} = $\text{dehors}_{t+1} \wedge (\text{parapluie}_{t+1} \leftrightarrow \text{parapluie}_t) \wedge (\text{pluie}_{t+1} \leftrightarrow \text{pluie}_t) \wedge (\text{sec}_{t+1} \leftrightarrow \text{sec}_t \wedge (\text{parapluie}_t \vee \neg\, \text{pluie}_t))$.

Le *non-déterminisme* peut être exprimé de plusieurs façons, explorées indépendamment dans différents articles : (a) par l'utilisation d'effets complexes, comme α CAUSE $p \leftrightarrow q$, un choix qui est au cœur de la mise-à-jour des croyances, cf. paragraphe 12.4 ; (b) par l'utilisation de disjonctions d'effets (rappelant l'union non déterministe de la logique dynamique présentée au paragraphe 12.3.4), comme lancer-pièce CAUSE pile ou CAUSE $\neg$ pile ; (c) par l'utilisation de règles causales récursives, qui est plus technique et que nous n'illustrons pas ici. Des langages d'action (notamment le langage $\mathcal{C}$) rendent également possibles l'expression de la *concurrence*, tandis que d'autres autorisent l'expression des *actions épistémiques*, permettant ainsi de distinguer les faits des connaissances de l'agent ; ainsi, l'action de tester si le fluent f est vrai ou faux est représentée par la règle causale α CAUSE $\mathbf{K}$ f ou CAUSE $\mathbf{K}$ $\neg f$, où $\mathbf{K}$ est la modalité de connaissance de la logique épistémique S5 (voir notamment [35]).

Les opérateurs de progression et de régression s'appliquent directement dans ces langages. Un état de croyance, dans le modèle d'incertitude binaire, est un ensemble non vide d'états, et peut donc être représenté de façon compacte par une formule propositionnelle cohérente. La progression et la régression associent donc à une formule cohérente et une action, une autre formule (toujours cohérente dans le cas de la progression et de la régression faible). La progression d'une formule φ par une action α consiste d'abord à effectuer la conjonction de φ_t (exprimant que φ est vraie avant l'action) et de Σ_α, puis à oublier dans $\varphi_t \wedge \Sigma_\alpha$ toutes les variables f_t, i.e., à dériver la conséquence logique la plus forte de $\varphi_t \wedge \Sigma_\alpha$ qui soit indépendante des variables f_t (voir par exemple [42]). La régression faible se calcule de façon analogue : la régression faible de ψ par α est le résultat de l'oubli des variables f_{t+1} dans $\psi_{t+1} \wedge \Sigma_\alpha$. La régression forte de ψ par α s'obtient en calculant des conditions minimales garantissant que l'application de α conduira à un état satisfaisant ψ. Ainsi, dans l'exemple précédent, la progression de sec $\wedge$ parapluie par sortir est (à l'équivalence logique près) dehors $\wedge$ parapluie $\wedge$ sec, et la progression de $\neg$ parapluie par sortir est dehors $\wedge$ $\neg$ parapluie $\wedge$ (pluie $\rightarrow$ $\neg$ sec), tandis que la régression faible de sec $\wedge$ pluie par sortir est parapluie $\wedge$ pluie $\wedge$ sec. On trouvera dans [43] une discussion sur le calcul et la complexité de ces opérations.

12.3.4 Logique dynamique

Il existe d'autres voies possibles pour résoudre les problèmes de représentation des actions. La *logique dynamique* est un formalisme initialement conçu en informatique théorique pour raisonner sur l'exécution des programmes. En plus des opérateurs booléens, elle dispose d'*opérateurs modaux* de la forme $[\alpha]$, où α est un programme. La combinaison d'un tel opérateur avec une formule donne une formule de la forme $[\alpha]\phi$ qui se lit « ϕ est vrai après toute exécution de α ». Au lieu d'un programme, on peut supposer que α est un événement ou une action. Par exemple l'action de basculer l'interrupteur a peut être décrite par les lois d'effet $(\neg U_a \rightarrow [B_a]U_a)$ et $(U_a \rightarrow [B_a]\neg U_a)$.

Dans le cadre de la logique dynamique, un aspect important du raisonnement sur les actions a été abordé dans [38] et porte sur la cohérence de la description d'un domaine. Il est montré que pour des langages d'action expressifs, au-delà de la cohérence logique, une bonne description d'un domaine doit être modulaire, dans le sens où les lois d'effet (décrivant les actions) ne doivent pas permettre de déduire de nouvelles lois statiques. Par exemple, les lois d'effet $P_1 \rightarrow [A]Q$, $P_2 \rightarrow [A]\neg Q$ et $\neg[A]\bot$ impliquent la loi statique $\neg(P_1 \wedge P_2)$; si cette loi n'est pas déjà déductible à partir des seules lois statiques alors les lois d'effet doivent être considérées comme problématiques.

Contrairement au calcul des situations, les états ne sont pas explicites en logique dynamique. Cette logique ne dispose pas non plus de quantification sur les actions, élément-clé de la solution de Reiter selon lui-même. Il a cependant été montré que la solution de Reiter peut être reproduite en logique dynamique pour le cas assez général de SSA explicites [70]. Dans de tels SSA, x doit être la seule variable d'action de $\gamma_P(x,s)$ et si une constante d'action A n'apparaît pas dans $\gamma_P(x,s)$ alors $\gamma_P(A,s)$ doit être logiquement équivalent à $P(s)$. Ces conditions sont naturelles pour un système satisfaisant l'inertie. Un exemple de SSA ne les satisfaisant pas serait le suivant : $\forall s(\forall x P(do(x,s))) \leftrightarrow \neg P(s))$ qui signifie que P est changé par toute action dans tout état (donc P est un fluent non inerte). Notons que la formule $\gamma_{U_a}(x,s)$ dans notre exemple (du paragraphe 12.3.2) satisfait ces conditions. Afin de traduire ces SSA en logique dynamique, on introduit des actions d'affectation de la forme $P := \phi$; une telle affectation décrit une action où P prend la valeur de ϕ à l'état précédent. Ceci permet de faire correspondre à chaque constante d'action A l'ensemble suivant d'affectations :

$$\sigma_{SSA}(A) \quad = \quad \{P := simp(\gamma_P(A)) \mid P \text{ apparaît dans } \gamma_P(x)\}$$

où $simp(\gamma_P(A))$ est obtenu à partir de $\gamma_P(x)$ par élimination de l'argument s, substitution de x par A et simplification des égalités. Dans notre exemple (du paragraphe 12.3.2), nous obtenons par la substitution de x par B_a :

$$\sigma_{SSA}(B_a) \quad = \quad \{U_a := (\neg U_a \wedge B_a{=}B_a) \vee (U_a \wedge B_a{\neq}B_a)\}$$

ce qui peut ensuite être simplifié en

$$\sigma_{SSA}(B_a) \quad = \quad \{U_a := \neg U_a\}.$$

Chaque occurrence d'un symbole d'action abstrait A est remplacé par l'affectation afférente. Comme il est montré dans [70] ceci constitue une solution (dans le sens de Reiter) au problème du décor. La solution de Reiter se trouve ainsi transférée à la logique dynamique, sans cependant faire appel à la quantification sur les actions. Il y est également montré que la solution de Reiter peut être combinée avec la logique épistémique, faisant ainsi le lien avec les logiques épistémico-dynamiques (cf. chapitre 2).

12.3.5 Réseaux bayésiens dynamiques

Un *réseau bayésien dynamique* est un réseau bayésien (cf. chapitre 8), dans lequel les variables sont considérées à différents instants. Pour chaque instant t, il existe un réseau bayésien reliant les variables considérées à l'instant t. De plus, entre chacun des réseaux instantanés composant le réseau bayésien dynamique, les seuls arcs autorisés sont ceux qui sont orientés vers le futur. Le graphe dirigé acyclique (DAG) temporel donné à la figure 12.1, muni des probabilités pour chaque variable, à chaque instant conditionnellement à chaque variable parente représente un réseau bayésien dynamique.

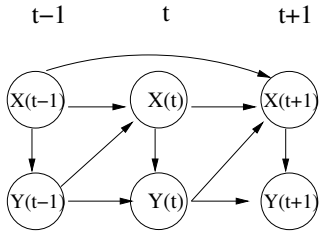


FIGURE 12.1 – Le DAG d'un réseau bayésien dynamique

Si l'on dispose de variables markoviennes, les deux probabilités suivantes déterminent complètement le comportement du système : la loi de probabilité des variables X_n et la loi de probabilité conditionnelle de X_{n+1}, lorsque X_n est donnée. L'hypothèse markovienne est faite le plus souvent, car elle permet des simplifications et parce qu'elle correspond à beaucoup de systèmes physiques. Un DAG temporel markovien ne peut pas admettre d'arc reliant des variables distantes de plus d'un intervalle de temps : en supprimant l'arc entre $X(t-1)$ et $X(t+1)$ le diagramme de l'exemple ci-dessus devient markovien, et une description restreinte aux instants t et $t+1$ suffit.

La valeur de vérité d'un fluent f à un instant donné t peut dépendre de sa valeur à un instant antérieur $t-\Delta$, ce qui se traduit en termes de probabilités par « l'équation de survie » suivante :

$$Pr(f(t)) = Pr(f(t) \mid f(t-\Delta)).Pr(f(t-\Delta)) + Pr(f(t) \mid \neg f(t-\Delta)).Pr(\neg f(t-\Delta))$$

La probabilité conditionnelle $Pr(f(t) \mid f(t-\Delta))$ est appelée *fonction de survie* (Syski 1979). La fonction de survie représente la tendance des propositions à persister étant donnés tous les événements qui peuvent les rendre fausses. Une fonction de survie classique est : $Pr(f(t) \mid f(t-\Delta)) = \exp^{-\lambda.\Delta}$, elle indique que la probabilité que f persiste décroît, depuis le dernier instant où f a été observé à une vitesse exponentielle déterminée par λ.

Si l'on possède des informations concernant des événements qui peuvent affecter la valeur de vérité du fluent, alors l'équation de survie est inadéquate. Généralement, la probabilité qu'une proposition f soit vraie en t est fonction

- de la probabilité qu'elle soit vraie en $t-\Delta$: $Pr(f(t-\Delta))$
- et de la probabilité qu'elle soit fausse en $t-\Delta$: $Pr(\neg f(t-\Delta))$
- et de la probabilité qu'il arrive un événement qui rend f vraie en t : $Pr(do(f,t))$
- et de la probabilité qu'il arrive un événement qui rend f faux en t : $Pr(do(\neg f,t))$.

Le problème de ce type d'approche probabiliste du raisonnement sur le changement, utilisée également par [32] et par [57] est qu'il nécessite la connaissance de beaucoup de probabilités *a priori*, même s'il n'est pas forcément nécessaire de résoudre tout le réseau probabiliste pour déterminer la probabilité d'une proposition et qu'on peut se limiter à un certain nombre d'instants intéressants. L'utilisation d'un réseau dynamique possibiliste permet de pouvoir raisonner sans avoir à connaître précisément ces probabilités [33]. Du point de vue de l'expressivité, un des intérêts de ce type d'approches est qu'elle permet d'exprimer une incertitude graduelle quantitative sur les croyances, les observations, et les lois causales.

12.4 Raisonnement sur le changement : la mise-à-jour

La *mise-à-jour* est un domaine de recherche qui se place à l'intersection du raisonnement sur l'action (d'où sa présence dans ce chapitre) et du changement des croyances (cf. chapitre 11). C'est un processus qui permet d'intégrer à une base de croyances un changement de l'état du système explicitement spécifié par une formule propositionnelle. Plus précisément, étant donnés un état de croyance K et une formule propositionnelle α, la mise-à-jour de K par α est la progression de K par une action, ou un événement exogène dont l'occurrence est connue, et dont l'effet est α. La mise-à-jour des croyances s'oppose à la révision des croyances où l'on intègre une nouvelle information sur le système à une base de croyances sur ce même système, en supposant qu'il n'a

pas évolué. Cette distinction a été mise au clair par Katsuno et Mendelzon [39], bien que la mise-à-jour fût antérieure à ces travaux [40, 71]. Ces premiers travaux sont issus en partie de chercheurs de la communauté « bases de données » (cf. chapitre III.3).

La distinction entre révision et mise-à-jour mérite qu'on s'y attarde un peu. Si la nouvelle information vient compléter la connaissance sur le monde, ce n'est pas le monde qui a évolué, mais seulement les connaissances de l'agent (il peut s'agir de la remise en cause d'une information erronée sur le monde, ou d'une information nouvelle concernant les caractéristiques du monde) ; l'opération de changement des croyances correspondante est alors une *révision* [10]. Cette révision est un simple ajout (ou *expansion*) lorsque l'information est cohérente avec les connaissances ; en revanche, en cas d'incohérence, la révision sélectionne les croyances que l'on doit rejeter afin de restaurer la cohérence (cf. chapitre 11). Si la nouvelle information caractérise une évolution explicite du monde (c'est-à-dire l'effet d'une action ou d'un événement exogène), alors on parle de *mise-à-jour*. La base de croyances mise à jour décrit le monde après son évolution, la mise-à-jour correspond donc à une *progression*.

Cette différence est illustrée sur l'exemple suivant [55]. Supposons que l'on dispose d'un panier qui contient soit une pomme soit une banane. Si on apprend qu'il ne contient pas de banane alors nos connaissances doivent être révisées et on va déduire que le panier contient une pomme. Cependant, si, partant de la même connaissance initiale, on apprend que le monde a évolué de façon à ce qu'il n'y ait plus de banane (quelqu'un effectue l'action de prendre la banane du panier si elle y était), dans ce cas, il faut faire une mise-à-jour de nos connaissances, c'est-à-dire que, maintenant, soit le panier est vide soit il contient une pomme.

Comme pour la révision, il n'existe pas un opérateur de mise-à-jour qui convienne en toute circonstance. Un aspect intéressant du problème est de déterminer les critères qui définissent les opérateurs « rationnels » de mise-à-jour, c'est-à-dire l'ensemble des postulats que doit vérifier un tel opérateur pour être considéré comme un opérateur valable. De façon similaire aux postulats d'Alchourrón, Gärdenfors et Makinson (dits postulats AGM) [1] caractérisant les opérateurs de révision « rationnels », Winslett [72] a été la première à proposer des postulats pour la mise-à-jour. Ces postulats ont inspirés Katsuno et Mendelzon [39] qui ont pu proposer un nouvel ensemble de postulats liés, de façon similaire aux postulats AGM (cf. chapitre 11), à l'existence d'un ensemble de relations de pré-ordre sur les états du système, où chaque pré-ordre est associé à un état. Dans [39], ces auteurs utilisent, en la précisant, l'idée qu'avait émise Winslett : pour mettre à jour une base de croyance, on peut faire évoluer chacun des modèles de la base indépendemment. La nouveauté apportée par Katsuno et Mendelzon est que chaque modèle doit évoluer vers le plus « proche » (au sens du pré-ordre associé à ce modèle). Pour Katsuno et Mendelzon, un opérateur de mise-à-jour est une fonction $\diamond$ qui à partir d'une formule $\mathcal{K}$ qui code les connaissances dont on dispose sur le monde et d'une formule ϕ qui code l'information que l'on a sur l'évolution du monde, renvoie une nouvelle formule $\mathcal{K} \diamond \phi$. Les postulats qu'ils proposent pour caractériser les opérateurs $\diamond$ « rationnels » sont les suivants :

U1 $\mathcal{K} \diamond \varphi$ implique φ.
U2 Si $\mathcal{K}$ implique φ alors $(\mathcal{K} \diamond \varphi)$ est équivalent à $\mathcal{K}$.
U3 Si $\mathcal{K}$ et φ sont satisfaisables alors $\mathcal{K} \diamond \varphi$ est satisfaisable.
U4 Si $\mathcal{K}_1$ équivaut à $\mathcal{K}_2$ et φ_1 équivaut à φ_2 alors $\mathcal{K}_1 \diamond \varphi_1$ équivaut à $\mathcal{K}_2 \diamond \varphi_2$.
U5 $(\mathcal{K} \diamond \varphi) \wedge \psi$ implique $\mathcal{K} \diamond (\varphi \wedge \psi)$.
U6 Si $\mathcal{K} \diamond \varphi_1$ implique φ_2 et $\mathcal{K} \diamond \varphi_2$ implique φ_1 alors $\mathcal{K} \diamond \varphi_1$ équivaut à $\mathcal{K} \diamond \varphi_2$.
U7 Si $\mathcal{K}$ est complète alors $(\mathcal{K} \diamond \varphi_1) \wedge (\mathcal{K} \diamond \varphi_2)$ implique $\mathcal{K} \diamond (\varphi_1 \vee \varphi_2)$.
U8 $(\mathcal{K}_1 \vee \mathcal{K}_2) \diamond \varphi$ équivaut à $(\mathcal{K}_1 \diamond \varphi) \vee (\mathcal{K}_2 \diamond \varphi)$.
U9 Si $\mathcal{K}$ est complète et $(\mathcal{K} \diamond \varphi_1) \wedge \varphi_2$ est satisfaisable, alors $\mathcal{K} \diamond (\varphi_1 \wedge \varphi_2)$ implique $(\mathcal{K} \diamond \varphi_1) \wedge \varphi_2$.

10. Comme le montrent [22], la révision reste cependant pertinente même lorsque l'état de croyance initial et la nouvelle formule ne réfèrent pas au même instant, pourvu qu'une distinction syntaxique (via un étiquetage temporel, en anglais « *time-stamping* ») soit faite entre un fluent à un instant et le même fluent à un autre : ce qui est important pour la révision n'est pas que le monde soit statique, mais que les propositions utilisées pour décrire le monde soient statiques. Ceci explique aussi que l'extrapolation de croyance corresponde également à un processus de révision [18].

U1 stipule que ϕ est une information qui caractérise le monde après son évolution (c'est un des postulats de Winslett). U2 exprime que si dans l'ensemble des états du système avant la mise-à-jour l'information ϕ est déjà vraie alors le système n'évolue pas. C'est ce postulat qui implique que l'on doit toujours préférer l'inertie à l'évolution spontanée. Cet axiome n'est pas toujours souhaitable, puisqu'il interdit l'existence d'états transitoires, c'est-à-dire des états dans lesquels le système peut ne pas rester car il évolue immédiatement vers d'autres états. Par ailleurs, le postulat U2 n'est pas désirable dans les systèmes évolutifs ne modélisant pas systématiquement la persistance par défaut. U3 exprime que si l'on possède une représentation cohérente du système et de son évolution alors la mise-à-jour est toujours réalisable (c'est aussi un des postulats de Winslett). Cependant, il peut exister des systèmes dans lesquels il n'y a pas de transition réalisable entre deux états : par exemple, $\varphi = mort$ et $\alpha = vivant$, dans cette situation, on peux souhaiter que la mise à jour échoue. Cet axiome n'est donc pas toujours souhaitable. U8 (également défini par Winslett) signifie que la mise-à-jour est définie comme un opérateur de progression. U9 est une restriction de la réciproque de U5. Pour une critique plus détaillée des postulats, voir [16, 37].

Le théorème de représentation suivant lie ces postulats à l'existence d'un ensemble de relations de pré-ordre entre états du monde :

Théorème 1 (Katsuno, Mendelzon) *$\diamond$ satisfait U1, U2, U3, U4, U5, U8, U9*[11] *si et seulement si pour tout $\omega \in \Omega$ il existe un pré-ordre total $\leq_\omega$ tel que*
(1) $\forall \omega' \in \Omega, \quad \omega <_\omega \omega'$ ($\leq_\omega$ est « fidèle »),
(2) $Mod(\mathcal{K} \diamond \varphi) = \bigcup_{\omega \models \mathcal{K}} \{\omega' \models \varphi \text{ t.q. } \forall \omega'' \models \varphi, \omega' \leq_\omega \omega''\}$.

(1) signifie que pour chaque modèle ω de $\mathcal{K}$, les modèles de la mise-à-jour de ω par φ sont les modèles de φ les plus proches de ω au sens de $\leq_\omega$, et (2) que l'ensemble des modèles de la mise-à-jour de $\mathcal{K}$ par φ est la réunion des ensembles de modèles de la mise-à-jour de chaque modèle de $\mathcal{K}$ par φ (ceci découle directement du postulat U8).

De nombreux opérateurs de mise-à-jour ont été proposés dans la littérature. Pour les définir, en vertu du théorème ci-dessus, il suffit de définir une relation de pré-ordre total fidèle entre mondes pour chaque état du monde. En pratique, un tel ensemble de pré-ordres représente une façon de *minimiser les changements* lors de l'évolution du système. Par exemple, Winslett a défini la relation $\leq_\omega^{\text{PMA}}$ entre mondes, appelée PMA. Elle est basée sur la fonction $\text{diff}_{\text{PMA}}(\omega_1, \omega_2)$ (représentant l'ensemble des variables dont l'affectation diffère entre les deux mondes ω_1 et ω_2) : $\omega_1 \leq_\omega^{\text{PMA}} \omega_2 \quad \Leftrightarrow_{def} \quad \text{diff}_{\text{PMA}}(\omega_1, \omega) \subseteq \text{diff}_{\text{PMA}}(\omega_2, \omega)$. Cette relation est fidèle et permet donc de définir un opérateur de mise-à-jour. L'opérateur de mise-à-jour correspondant peut aussi être caractérisé en termes d'indépendance (toutes les conséquences logiques de $\mathcal{K}$ qui sont indépendantes de φ persistent) [50]. Sur l'exemple de [55], la connaissance initiale est $\mathcal{K} = (banane \wedge \neg pomme) \vee (pomme \wedge \neg banane)$ il y a donc deux modèles $\omega_1 = \{\neg pomme, banane\}$ et $\omega_2 = \{pomme, \neg banane\}$. On apprend ensuite ($\varphi$) que quelqu'un a enlevé la banane si elle y était (mise-à-jour par $\neg banane$), les états du système qui représentent l'information φ sont ω_2 et $\omega_3 = \{\neg pomme, \neg banane\}$. $\mathcal{K} \diamond_{\text{PMA}} \varphi$ peut se calculer comme l'union pour chaque modèle ω de $\mathcal{K}$, des plus proches modèles dans φ de ω. Ici, le plus proche modèle de ω_1 parmi ceux de φ pour la relation $\leq_\omega^{\text{PMA}}$ est ω_3, le plus proche modèle de ω_2 parmi ceux de φ est lui-même ($\leq_\omega^{\text{PMA}}$ est fidèle). Donc l'ensemble des modèles de $\mathcal{K} \diamond_{\text{PMA}} A$ est $\{\omega_2, \omega_3\}$. Ce qui signifie qu'après mise-à-jour, soit il y a seulement une pomme dans le panier, soit le panier est vide.

La relation PMA a été raffinée (PMA avec priorités [71]) en affectant des priorités à certains fluents : cela permet de considérer que les fluents ne persistent pas tous de la même manière. D'autres opérateurs de mise-à-jour vont plus loin dans l'expressivité, comme celui proposé par Cordier et Siegel [13] qui permet de prendre en compte des contraintes de transition (plus ou moins importantes) ; ces contraintes prennent la forme de couples de formules (φ, ψ) et sont satisfaites par un couple de mondes (ω, ω') lorsque ω satisfait φ et ω' satisfait ψ. ω' est alors considéré plus proche de ω que ω'' si la transition (ω, ω') viole moins de contraintes prioritaires que la transition (ω, ω'').

11. Si U6 et U7 sont utilisés à la place de U9 alors le théorème donne un pré-ordre fidèle qui n'est que partiel.

La mise-à-jour *à la* Katsuno et Mendelzon (et, en particulier, la PMA de Winslett [71]) est fondée sur la minimisation des changements. Cependant, la minimisation des changements n'est pas toujours souhaitable pour la mise-à-jour. En particulier, Herzig et Rifi [37] ont montré que les approches fondées sur la minimisation des changements traitent mal le problème de la mise-à-jour par une disjonction ; plus formellement, un opérateur de mise-à-jour qui satisfait les postulats de Katsuno et Mendelzon ne peut pas traiter correctement la disjonction ; le postulat qui en est responsable est U5.

Pour cette raison, plusieurs chercheurs ont étudié des opérateurs de mise-à-jour non fondés sur la minimisation, et en particulier, la famille d'opérateurs de mise-à-jour à base de *dépendance* : une telle mise-à-jour d'une base de croyances, représentée par une formule β, par une formule α consiste d'abord à oublier dans β « toutes les informations qui concernent » α (laissant inchangées les valeurs de vérité des variables qui ne sont pas touchées par la mise-à-jour), puis à joindre α au résultat. Il reste maintenant à préciser ce qu'il faut entendre par « les informations qui concernent la formule α » ; cette notion de pertinence est induite par une relation de dépendance entre formules : α concerne β si et seulement si β dépend de α ; la généralité de l'approche tient au fait que cette notion de dépendance entre formules peut varier.

La plupart des approches de la mise-à-jour à base de dépendance considèrent que cette relation de dépendance formule-formule est induite à partir d'une relation de dépendance entre formules et variables propositionnelles, qui est ensuite étendue à une dépendance formule-formule : α et β sont dépendantes si et seulement si il existe au moins une variable dont α et β dépendent. On trouve des exemples de tels opérateurs de mise-à-jour dans [34], [15] et [37]. Ce principe permet de corriger de nombreux aspects contre-intuitifs des approches fondées sur la minimisation, et de plus il conduit généralement à une complexité algorithmique moindre. Cependant, il a pour léger inconvénient d'être trop peu conservatif (il a tendance à oublier trop d'informations de la base de croyance initiale) ; pour remédier à cela, la dépendance entre formules et variables peut être remplacée par une dépendance entre formules et littéraux [36].

Puisque la mise-à-jour par une formule α peut être vue comme la progression par une action particulière dont l'effet est α (« rendre α vrai ») (voir une discussion dans [41]), il est pertinent de situer la mise-à-jour par rapport aux langages d'action propositionnels. On peut d'abord remarquer que STRIPS est un cas particulier des deux formalismes (et correspond à des mises-à-jour par des conjonctions de littéraux). L'axiome U8, qui exprime que la mise-à-jour d'un ensemble de modèles est l'union des mises-à-jour des modèles individuels, correspond à la définition même de la progression d'un état de croyance par une action. En revanche, les deux modèles diffèrent dans la variété des actions envisageables. D'un côté, la mise-à-jour offre la possibilité de prendre en compte des effets disjonctifs, reflétant un effet unique, mais imparfaitement connu, ou plus généralement des effets consistant en des formules propositionnelles complexes. D'un autre côté, les langages d'action permettent les effets conditionnels (si pile alors retourner-pièce cause ¬pile, si ¬ pile alors retourner-pièce cause pile), non déterministes comme lancer-pièce cause pile ou cause ¬pile), concurrents (si g et d sont les actions consistant à soulever le côté gauche et le côté droit d'une table, et si un verre d'eau est posé sur la table, alors g cause renversé, d cause renversé, $g||d$ cause $\top$), ainsi que des règles causales statiques permettant des ramifications. Les approches visant à unir les potentialités des deux approches ne sont pas légion. Quelques approches de la mise-à-jour prennent en compte la ramification via l'utilisation de contraintes d'intégrité [15], ou permettent les mises-à-jour non déterministes [10], conditionnelles ou concurrentes [36].

Si la mise-à-jour correspond à la progression d'action, il en existe une généralisation (appelée justement *mise-à-jour généralisée*) qui permet également la révision et l'abduction d'événements [9]. La mise-à-jour généralisée permet par exemple de traiter le scénario suivant : l'agent se réveille le matin et croit que la pelouse est sèche, comme la veille au soir. Il observe que la pelouse est mouillée, ce qui conduit d'abord à une révision des croyances, puis à une abduction d'événement (il a plu), et enfin une mise-à-jour par les effets de l'événement (la route est mouillée également). L'extrapolation de croyances [18] et les formalismes qui en sont proches, notamment [6], ne traitent, eux, que de l'abduction d'événements. Enfin, la mise-à-jour peut être vue comme une forme ordinale de l'opération d'*imaging* de Lewis [17] ainsi que de la phase prédictive du filtre de Kalman [14, 5].

Pouvoir réaliser à la fois des révisions et des mises-à-jour intéresse également Goldszmidt et Pearl [30] qui raisonnent sur un ensemble de règles par défaut (ces règles sont de type causal) représentant le système, en déduisent un ordre sur les couples d'états du monde qu'ils filtrent selon la nouvelle information. Si l'opération est une révision par φ à l'instant final alors les couples d'états où l'état final satisfait φ voient leur plausibilité croître. Dans le cadre d'une mise-à-jour par φ, on effectue une révision par l'action fictive $do(\varphi)$.

L'importance des travaux de Winslett et de Katsuno et Mendelzon se situe sur deux plans. Premièrement, l'établissement d'une distinction claire entre la révision et la mise-à-jour d'une base de croyance. Deuxièmement, l'élaboration d'un ensemble de postulats garantissant qu'une mise-à-jour rationnelle est liée à l'existence d'un ensemble de pré-ordres entre les états possibles du système. Katsuno & Mendelzon et Winslett se sont placés implicitement dans le cas particulier où les fluents sont par défaut inertes (ils ne changent que si une action ou un événement se produit pour les faire changer). De plus, une autre hypothèse implicite est faite à cause du postulat U3 (MB4 chez Winslett) : affirmer que toute mise-à-jour est réalisable signifie que l'on a toujours des informations cohérentes avec l'évolution du monde, c'est-à-dire que les connaissances sont correctes.

Des travaux se sont intéressés à l'extension de la mise à jour à des cadres plus expressifs :

- les ASP : Slota et Leite ont repris les postulats de Katsuno et Mendelzon pour les adapter aux programmes logiques [64]. Ces mêmes auteurs ont aussi étudié la définition d'opérateurs de mise-à-jour pour des bases de croyance hybrides [65, 63]. De telles bases hybrides sont constituées d'une composante ontologique (exprimée dans le langage des TBox et ABox des logiques de description) et d'une composante règles (exprimée dans le langage de la programmation logique). Plus précisément, le cadre de la première composante est la logique de description ALCIO (ALC avec inverse et nominaux) et le cadre de la seconde composante est la programmation par ensembles-réponses (ASP) sous la sémantique stable (cf. chapitre II.4).Des opérations de mise-à-jour sont étudiées plus particulièrement pour la sémantique de l'équivalence forte pour ASP ainsi que pour une proposition récente de bases de croyance hybrides de Motik et Rosati, à savoir la logique « *Minimal Knowledge and Negation as Failure* » avec son langage associé de programmes MKNF [56].
- les états de croyance : Lang, Marquis et Williams [44] définissent des opérateurs de mise-à-jour portant sur des états épistémiques, qui permettent d'exprimer en plus des croyances leurs plausibilités relatives. Ces états épistémiques sont représentés par des ordres sur les mondes. Les auteurs étendent la classe des opérateurs de mise à jour basés sur la dépendance aux états épistémiques. Baral et Zhang [2] généralisent la mise-à-jour de telle façon qu'elle soit définie sur un language qui permet la distinction entre faits et connaissances, comme la logique épistémique S5 ; ce processus appelé *knowledge update* permet de traiter les effets des actions épistémiques en *mettant à jour* les formules épistémiques écrivant les connaissances de l'agent. Un tel cadre prend le point de vue d'un agent modélisateur O qui raisonne sur l'état des croyances d'un autre agent ag. Ainsi, par exemple, la mise-à-jour d'un modèle de S5 par $K_{ag}\varphi$ signifie que O met à jour ses croyances sur les connaissances de ag ; l'état mental de ag est vu comme une partie du monde extérieur vu de l'agent O, et la mise-à-jour par $K_{ag}\varphi$ correspond à une action dont l'effet est de rendre $K_{ag}\varphi$ vrai (par exemple, l'action de dire à ag que φ est vraie).
- les logiques de description (cf. chapitre 5) : Liu et al. [49] ont étudié des opérateurs de mise-à-jour pour les assertions d'une base de connaissances (la « ABox »). Ils ont mis en évidence un problème d'expressivité qui surgit dans ce cadre : parfois le résultat attendu d'une mise-à-jour ne peut pas être codé par une assertion de la logique de description de base ALC. Par exemple, l'assertion

  $$\texttt{marie} : \texttt{Personne} \sqcap \exists \texttt{enfant\_de}.\texttt{Personne} \sqcap \forall \texttt{enfant\_de}.(\texttt{Personne} \sqcap \texttt{Heureux})$$

  exprime que tous les enfants de Marie sont heureux. Si on met à jour la ABox contenant ce fait par le fait que Pierre est devenu malheureux, c'est-à-dire par l'assertion `pierre` :

$\texttt{Personne} \sqcap \neg\texttt{Heureux}$, alors il faut prendre en compte deux cas possibles : le cas où Pierre est un des enfants de Marie et le cas où il ne l'est pas. Intuitivement, le principe de changement minimal requiert alors que le résultat de la mise à jour est d'une part la nouvelle information (Pierre est malheureux) et d'autre part le fait que chaque enfant de Marie a ou bien la propriété d'être heureux, ou bien s'appelle Pierre. Cette dernière assertion ne peut pas être exprimé en ALC : il requiert son extension par des noms d'objets. La logique ainsi étendue (appelée ALCO) permet d'écrire $\texttt{marie} : \texttt{Personne} \sqcap \exists\texttt{enfant\_de}.\texttt{Personne} \sqcap \forall\texttt{enfant\_de}.(\texttt{Personne} \sqcap (\texttt{Heureux} \sqcup \{\texttt{pierre}\}))$. Il s'avère que presque toutes les logiques de description posent des problèmes d'expressivité similaires. Admettre des noms d'objets comme concepts efface de fait la distinction entre les assertions de l'ABox (qui, elles, parlent des objets) et des inclusions de concepts de la TBox qui ne devraient pas parler des objets. Ce n'est pas très satisfaisant car cette distinction est une des idées de base des logiques de description.

- les descriptions d'action : Eiter et al. [19] définissent un cadre pour le changement minimal de descriptions d'actions, qu'ils appellent "action description updates", bien qu'il ne soit pas entièrement clair que le processus ne s'apparente pas plutôt à une révision.
- l'argumentation abstraite : la mise à jour et le changement en général intéressent aussi les chercheurs de ce domaine (cf. chapitre 10). Un système d'argumentation abstrait étant représenté par un graphe dont les sommets sont des arguments et les arcs des attaques entre eux, certains auteurs [8, 7, 12, 45] s'intéressent à l'impact d'un changement (par ajout/retrait d'arguments ou d'attaques) sur ce système. Baumann et Brewka [4, 3] ont introduit la notion d'« *enforcement* », qui est très similaire à la notion de mise-à-jour puisque l'idée est de modifier de façon minimale un système d'argumentation afin qu'il satisfasse un but donné (le plus souvent exprimé en termes d'arguments qui doivent être acceptés). Ils proposent une relation de préférence entre système d'argumentation définie de façon similaire aux relations de préférence entre modèles dans la mise-à-jour classique.

12.5 Conclusion

Le raisonnement sur l'action et le changement est l'un des domaines les plus anciens de l'intelligence artificielle. Depuis 1995, la problématique fait l'objet d'un workshop bisannuel *International Workshop on Nonmonotonic Reasoning, Action and Change (NRAC)* qui se tient en conjonction avec la conférence IJCAI (*International Joint Conference on Artificial Intelligence*).

Plusieurs époques se sont succédées, durant lesquelles les chercheurs se sont intéressés à plusieurs cadres formels pour la modélisation des principaux processus du raisonnement sur l'action : STRIPS d'abord, puis les approches fondées sur la minimisation des changements, puis les approches fondées sur les *successor state axioms*. À cette variété de cadres formels correspond une variété de langages utilisés : logique propositionnelle, calcul des situations, logique dynamique, langages graphiques (entre autres).

Le raisonnement sur l'action et le changement entretient des liens étroits avec d'autres domaines de l'intelligence artificielle, notamment le raisonnement non monotone, le changement des croyances, le raisonnement dans l'incertain, la planification (et en particulier les processus décisionnels de Markov) ; il entretient également des liens avec l'automatique (filtre de Kalman, systèmes à événements discrets).

Références

[1] C. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change : partial meet contraction and revision functions. *J. of Symbolic Logic*, 50 :510–530, 1985.

[2] C. Baral and Y. Zhang. Knowledge updates : Semantics and complexity issues. *Artif. Intell.*, 164(1-2) :209–243, 2005.

[3] R. Baumann. What Does it Take to Enforce an Argument ? Minimal Change in Abstract Argumentation. In *Proc. European Conf. on Artificial Inteligence (ECAI'12)*, pages 127–132, 2012.

[4] R. Baumann and G. Brewka. Expanding Argumentation Frameworks : Enforcing and Monotonicity Results. In *Proc. Int. Conf. on Computational Models of Arugement (COMMA'10)*, pages 75–86, 2010.

[5] S. Benferhat, D. Dubois, and H. Prade. Kalman-like filtering in a possibilistic setting. In *Proc. European Conf. on Artificial Intelligence (ECAI'00)*, pages 8–12, 2000.

[6] S. Berger, D. J. Lehmann, and K. Schlechta. Preferred history semantics for iterated updates. *J. of Logic and Computation*, 9(6) :817–833, 1999.

[7] G. Boella, S. Kaci, and L. van der Torre. Dynamics in argumentation with single extensions : Abstraction principles and the grounded extension. In *Proc. European Conf. on Symbolic and Qualitative Aspects of Reasoning under Uncertainty (ECSQARU'09)*, pages 107–118, 2009.

[8] G. Boella, S. Kaci, and L. van der Torre. Dynamics in argumentation with single extensions : Attack refinement and the grounded extension. In *Proc. Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS'09)*, pages 1213–1214, 2009.

[9] C. Boutilier. A unified model of qualitative belief change : A dynamical systems perspective. *Artificial Intelligence*, 98(1-2) :281–316, 1998.

[10] G. Brewka and J. Hertzberg. How to do things with worlds : On formalizing actions and plans. *J. of Logic and Computation*, 3(5) :517–532, 1993.

[11] M. Castillo, O. Gasquet, and A. Herzig. Formalizing action and change in modal logic I : The frame problem. *J. of Logic and Computation*, 9(5) :701–735, 1999.

[12] C. Cayrol, F. Dupin de Saint-Cyr, and M.-C. Lagasquie-Schiex. Change in abstract argumentation frameworks : Adding an argument. *J. of Artificial Intelligence Research*, 38 :49–84, 2010.

[13] M.-O. Cordier and P. Siegel. Prioritized transitions for updates. In *Proc. European Conf. on Symbolic and Qualitative Aspects of Reasoning under Uncertainty (ECSQARU'95)*, pages 142–150, 1995.

[14] C. Cossart and C. Tessier. Filtering vs. revision and update : Let us debate ! In *Proc. European Conf. on Symbolic and Qualitative Aspects of Reasoning under Uncertainty (ECSQARU'99)*, pages 116–127, 1999.

[15] P. Doherty, W. Lukaszewicz, and E. Madalinska-Bugaj. The PMA and relativizing minimal change for action update. In *Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 258–269, 1998.

[16] D. Dubois, F. Dupin de Saint-Cyr, and H. Prade. Update postulates without inertia. In *Proc. European Conf. on Symbolic and Qualitative Aspects of Reasoning under Uncertainty (ECSQARU'95)*, pages 162–170, 1995.

[17] D. Dubois and H. Prade. Belief revision and updates in numerical formalisms : An overview, with new results for the possibilistic framework. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI'93)*, pages 620–625, 1993.

[18] F. Dupin de Saint-Cyr and J. Lang. Belief extrapolation (or how to reason about observations and unpredicted change). *Artificial Intelligence*, 175(2) :760–790, 2011.

[19] T. Eiter, E. Erdem, M. Fink, and J. Senko. Updating action domain descriptions. *Artif. Intell.*, 174(15) :1172–1221, 2010.

[20] R.E. Fikes and N.J. Nilsson. Strips : A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2 :189–208, 1971.

[21] J.J. Finger. *Exploiting constraints in design synthesis.* PhD thesis, Stanford University, Stanford, CA, 1987.

[22] N. Friedman and J. Y. Halpern. Modeling belief in dynamic systems, part II : Revision and update. *J. of Artificial Intelligence Research*, 10 :117–167, 1999.

[23] N. Friedman and J.Y. Halpern. A knowledge based framework for belief change part II : revision and update. In *Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'94)*, pages 190–201, 1994.

[24] H. Geffner. Causal theories for nonmonotonic reasoning. In *Proc. National Conf. on Artificial Intelligence (AAAI'90)*, pages 524–530, 1990.

[25] M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *J. of Logic Programming*, 17 :301–321, 1993.

[26] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, C. Weld, and D. Wilkins. The planning domain definition language. Technical report, AIPS-98 Planning Competition, 1998.

[27] L. Giordano, A. Martelli, and C. Schwind. Dealing with concurrent actions in modal action logics. In *Proc. European Conf. on Artificial Intelligence (ECAI'98)*, pages 537–541, 1998.

[28] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153 :49–104, 2004.

[29] E. Giunchiglia and V. Lifschitz. An action language based on causal explanation : Preliminary report. In *Proc. National Conf. on Artificial Intelligence (AAAI'98)*, pages 623–630, 1998.

[30] M. Goldszmidt and J. Pearl. Rank-based systems : A simple approach to belief revision, belief update, and reasoning about evidence and actions. In *Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 661–672, 1992.

[31] S. Hanks and D. McDermott. Default reasoning, nonmonotonic logics, and the frame problem. In *Proc. National Conf. on Artificial Intelligence (AAAI'86)*, pages 328–333, 1986.

[32] S. Hanks and D. McDermott. Modelling and uncertain world i : symbolic and probabilistic reasoning about change. *Artificial Intelligence*, 66 :1–55, 1994.

[33] A. Heni, N. Ben Amor, S. Benferhat, and A. Alimi. Dynamic possibilistic networks : Representation and exact inference. In *Proc. IEEE Int. Conf. on Computational Intelligence for Measurement Systems and Applications (CIMSA'07)*, pages 1–8, 2007.

[34] A. Herzig. The PMA revisited. In *Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'96)*, pages 40–50, 1996.

[35] A. Herzig, J. Lang, and P. Marquis. Action representation and partially observable planning using epistemic logic. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI'03)*, pages 1067–1072, 2003.

[36] A. Herzig, J. Lang, P. Marquis, and Th. Polacsek. Updates, actions, and planning. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI'01)*, pages 119–124, 2001.

[37] A. Herzig and O. Rifi. Propositional belief base update and minimal change. *Artificial Intelligence*, 115(1) :107–138, 1999.

[38] A. Herzig and I. J. Varzinczak. Metatheory of actions : beyond consistency. *Artificial Intelligence*, 171 :951–984, 2007.

[39] H. Katsuno and A.O. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 387–394, 1991.

[40] A.M. Keller and M. Winslett. On the use of an extended relational model to handle changing incomplete information. In *IEEE Trans. on Software Engineering*, volume SE-11 :7, pages 620–633, 1985.

[41] J. Lang. Belief Update Revisited. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI'07)*, pages 2517–2522, 2007.

[42] J. Lang, P. Liberatore, and P. Marquis. Propositional independence : Formula-variable independence and forgetting. *J. of Artificial Intelligence Research*, 18 :391–443, 2003.

[43] J. Lang, F. Lin, and P. Marquis. Causal theories of action : a computational core. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI'03)*, pages 1073–1078, 2003.

[44] J. Lang, P. Marquis, and M-A. Williams. Updating epistemic states. In *Proc. Australian Joint Conf. on Artificial Intelligence (AI'01)*, pages 297–308, 2001.

[45] Beishui Liao, Li Jin, and Robert C. Koons. Dynamics of argumentation systems : A division-based method. *Artificial Intelligence*, 175(11) :1790 – 1814, 2011.

[46] V. Lifschitz. Frames in the space of situations. *Artificial Intelligence*, 46 :365–376, 1990.

[47] V. Lifschitz and A. Rabinov. Things that change by themselves. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI'89)*, pages 864–867, 1989.

[48] F. Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI'95)*, pages 1985–1991, 1995.

[49] Hongkai Liu, Carsten Lutz, Maja Milicic, and Frank Wolter. Foundations of instance level updates in expressive description logics. *Artificial Intelligence*, 175(18) :2170–2197, 2011.

[50] P. Marquis. Possible models approach via independency. In *Proc. European Conf. on Artificial Intelligence (ECAI'94)*, pages 336–340, 1994.

[51] J. McCarthy. Epistemological problems of artificial intelligence. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI'77)*, pages 1038–1044, 1977.

[52] J. McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28(1) :1038–1044, 1986.

[53] J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, volume 4, pages 463–502. 1969.

[54] Y. Moinard. Note about cardinality-based circumscription. *Artificial Intelligence*, 119(1-2) :259–273, 2000.

[55] M. Morreau. Planning from first principles. In *Belief revision*, pages 204–219. Cambridge University Press, 1992.

[56] B. Motik and R. Rosati. Reconciling description logics and rules. *J. of the Association for Computing Machinery*, 57(5), 2010.

[57] J. Pearl. Embracing causality in formal reasoning. *Artificial Intelligence*, 35 :259–271, 1988.

[58] E. Pednault. Adl : Exploring the middle ground between strips and the situation calculus. In *Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 324–332, 1989.

[59] R. Reiter. The frame problem in the situation calculus : A simple solution (sometimes) and a completeness result for goal regression. In *Artificial Intelligence and Mathematical Theory of Computation : Papers in Honor of John McCarthy*, pages 359–380. Academic Press, 1991.

[60] E. Sandewall. *Features and Fluents*. Oxford University Press, 1995.

[61] R. Scherl and H. J. Levesque. The frame problem and knowledge producing actions. *Artificial Intelligence*, 144(1-2), 2003.

[62] Y. Shoham. *Reasoning about Change - Time and Causation from the Standpoint of Artificial Intelligence*. MIT Press, 1988.

[63] M. Slota. *Updates of Hybrid Knowledge bases*. PhD thesis, Universidade Nove de Lisboa, 2012.

[64] M. Slota and J. Leite. On semantic update operators for answer-set programs. In *Proc. European Conf. on Artificial Intelligence*, pages 957–962, 2010.

[65] M. Slota, J. Leite, and T. Swift. Splitting and updating hybrid knowledge bases. *Theory and Practice of Logic Programming*, 11(4-5) :801–819, 2011.

[66] L.A. Stein and L. Morgenstern. Motivated action theory : A formal theory of causal reasoning. *Artificial Intelligence*, 71(1) :1–42, 1994.

[67] M. Thielscher. The logic of dynamic systems. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI'95)*, pages 1956–1962, 1995.

[68] M. Thielscher. Ramification and causality. *Artificial Intelligence*, 89(1–2) :317–364, 1997.

[69] H. Turner. A logic of universal causation. *Artificial Intelligence*, 113(1-2) :87–123, 1999.

[70] H. van Ditmarsch, A. Herzig, and T. de Lima. From Situation Calculus to Dynamic Epistemic Logic. *J. of Logic and Computation*, 21(2) :179–204, 2011.

[71] M. Winslett. Reasoning about action using a possible models approach. In *Proc. National Conf. on Artificial Intelligence (AAAI'88)*, pages 89–93, 1988.

[72] M. Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.