Techniques d'abstraction dans la vérification des systèmes concurrents

–Résumé de thèse –

par Constantin Enea

Superviseurs:

Prof.dr. Ferucio Laurentiu Tiplea
Le département d'informatique
Université "Al. I. Cuza"
Iasi, Romania

Prof.dr. Anatol Slissenko
Laboratoire d'Algorithmique, Complexité et Logique
Université Paris 12, Val de Marne
Paris, France

Paris, 2007

1 Introduction

Comme les systèmes matériels et logiciels grandissent de façon continue en échelle et fonctionnalités, la probabilité d'erreurs subtiles devient toujours plus grande. Dans l'industrie, le testage a été traditionnellement la technique du dépannage principale. Par exemple, "bêta distributions" d'un logiciel sont envoyées dehors à un groupe de gens disposés à l'utiliser et à faire un rapport sur les erreurs rencontrées. D'autres programmes, comme le code du microprocesseur réalisé dans le matériel, sont souvent testés automatiquement en nourrissant des séquences d'entrées et en comparant les sorties obtenues aux sorties désirées. La validité du testage est basée sur la recherche exhaustive pour les entrées possibles (lesquelles peuvent être impraticables en grand nombre) et on ne peut l'appliquer aux *systèmes réactifs* qui maintiennent une interaction continue avec leur environnement. Les systèmes réactifs sont mis en contraste avec la vision démodée d'un programme comme quelque chose qui enregistre des entrées, il calcule pendant quelques temps, il produit un résultat et il termine.

L'usage des *méthodes formelles* a été proposé pour vaincre ces problèmes. Les méthodes formelles couvrent toutes les approches pour la spécification et la vérification basées sur des formalismes mathématiques. Ils contiennent trois parties: un modèle mathématique du système, un langage formel pour exprimer la spécification et une méthodologie pour établir si le modèle du système satisfait la spécification.

Les méthodes formelles peuvent être classées comme *syntactiques* et *sémantiques*. Dans les méthodes syntactiques, le système est décrit dans un langage de programmation dont les constructions élémentaires sont exprimées par des axiomes et les constructions plus grandes par des règles d'inférence dans un système de preuve. La spécification est donnée dans un langage formel puissant et la preuve d'exactitude se réduit à une preuve dans ce système. Dans les méthodes sémantiques, le modèle d'un programme consiste dans une description de tous ses comportements dans une structure mathématique comme un système de transitions. La spécification est une formule dans une logique qui est interprétée sur de telles structures (e.g. la logique temporelle) et l'exactitude est prouvée en montrant que la formule est satisfaite par le modèle.

Un exemple d'une méthode formelle sémantique est *la vérification énumérative* (en anglais, model checking) développée indépendamment par Clarke et Emerson [19] aux États-Unis et par Quielle et Sifakis [82] en France. Dans cette approche, les systèmes sont modelés par des structures Kripke et les spécifications sont exprimées dans une logique temporelle propositionnelle. Une procédure de recherche efficace est utilisée pour déterminer si

la spécification est satisfaite dans la structure Kripke. Les plus importants avantages de la vérification énumérative sont qu'elle est complètement automatique et dans les cas des réponses négatives elle fournit un contre-exemple qui montre pourquoi la spécification n'est pas satisfaite.

L'inconvénient principal de la vérification énumérative est *l'explosion des états*, un problème qui peut apparaître dans des systèmes qui consistent dans un ensemble de processus concourants qui fonctionnent en parallèle. L'ensemble des comportements possibles du système contient une séquence pour chaque imbrication d'actions des composants et il peut grandir exponentiellement dans le nombre de composants. La présence des données peut aussi contribuer à ce problème. Un bit supplémentaire utilisé par un programme peut doubler le nombres d'états. Beaucoup de solutions pour ce problème ont été proposées. Elles essaient d'améliorer la représentation de l'espace d'états ou de le réduire en ignorant certains détails. Dans la première catégorie on peut trouver *la vérification énumérative symbolique* [13, 72] qui représente l'espace d'états par des diagrammes de décision binaires [11] et qui a produit des résultats spectaculaires [13, 21, 75], ou *la vérification énumérative "on-the-fly"* [29, 42] qui étend progressivement l'espace d'états du système. De la deuxième catégorie, on peut mentionner les techniques basées sur des *ordres partiels* [47, 80, 91], des techniques basées sur des *symétries* [20, 36, 58], *la modularisation* [26, 49, 50], ou *l'abstraction* [30, 59, 60, 66, 57, 14, 89, 81, 23, 93, 31, 48, 84, 33, 73, 4, 92, 85].

Les techniques d'abstraction, souvent basées sur l'interprétation abstraite de Cousot [30], fournissent une méthode pour exécuter symboliquement les systèmes en utilisant le domaine abstrait à la place du domaine concret. Par exemple, *l'abstraction des données* de [23] part d'une fonction d'abstraction dont le domaine est l'ensemble de données effectives du système et le codomaine est un petit ensemble de données abstraites. Cette fonction est étendue aux états et aux transitions pour obtenir une version abstraite du système considéré. *"Shape analysis"* [79, 85] est une technique d'analyse du flux des données, utilisée principalement pour l'analyse des données allouées dynamiquement. Les états possibles de la mémoire, qui survient à un moment donné dans le programme, sont représentés par des "shape graphs". Dans un tel graphe, les noeuds représentent des cellules d'espace mémoire ou des ensembles des cellules "indiscernables". *L'abstraction par prédicats* est une autre technique d'abstraction proéminente [48, 33, 92, 4]. L'idée principale est d'attribuer aux objets concrets (états d'un système de transitions, données d'un type de données, etc.) des objets abstraits selon leur évaluation sous un ensemble fini de prédicats. Dans [6], Bidoit et Boisseau emploient *des abstractions algébriques* pour vérifier des propriétés du protocoles de sécurité modelés par des algèbres universelles. Leur abstraction est basée sur des ho-

momorphismes et la technique de *dupliquer les symboles prédicatifs* [23, 32].

Afin qu'une abstraction soit utile, elle doit *préserver des propriétés*. Les formes de préservation des propriétés, qui sont étudiées principalement dans la littérature, impliquent seulement des logiques sous l'interprétation binaire classique. Les logiques multi-valuées fournissent une alternative à la logique booléenne classique pour modeler et raisonner au sujet des systèmes. En ajoutant de nouvelles valeurs de vérité, l'incertitude et le désaccord peuvent être modelés explicitement et beaucoup d'applications ont été trouvées dans les bases de données [45], dans la représentation de la connaissance [46] ou dans la vérification du matériel et du logiciel [8, 9, 55].

Pour la vérification du matériel, des outils de simulation et des réalisations des circuits multi-valués véritables ont été proposés, les risques dynamiques ont été modelés en introduisant des états faux pour trouver des régions chevauchantes des signaux en concurrence [12], etc. Pour la vérification du logiciel on a besoin d'incertitude parce qu'on ne peut savoir si certains comportements devraient être possibles, on a besoin du désaccord parce que l'on peut avoir des acteurs différents qui sont en désaccord pour la manière dont les systèmes devraient se comporter. On a besoin aussi de représenter l'importance relative parce que quelques comportements sont essentiels et d'autres peuvent ou pas être rendus effectifs. Des techniques de *vérification énumérative multi-valuée* ont été proposées par beaucoup de chercheurs [15, 62, 17, 16, 18, 51, 10, 63, 64], en motivant, encore plus, l'usage des logiques multi-valuées dans le processus de la vérification.

2 Logiques multi-valuées

Cette thèse traite des abstractions qui préservent des propriétés exprimées dans des logiques sous des interprétations multi-valuées. On commence par présenter le concept d'*algèbre de vérité* qui est le fondement des interprétations multi-valuées.

Définition 2.1 Une *algèbre de vérité* est un uplet $\mathcal{B} = (B, \wedge, \vee, \neg)$, où:

- $(B, \leq)$ est un treillis complet dont $\leq$ est une ordre binaire donnée par $a \leq b$ si et seulement si $a \vee b = b$, pour tout a,$b \in B$;
- $\wedge$ est la plus grande borne inférieure et $\vee$ est la plus petite borne supérieure;
- $\neg : B \to B$ est une bijection pour laquelle $\neg 0 = 1$ et $\neg 1 = 0$.

La Figure 1 contient différentes algèbres de vérité: la logique classique 2-valuée est représentée dans la Figure 1(a), la logique 3-valuée Kleene [61]

de la Figure 1(b) ajoute la valeur de vérité $\perp$ pour les valeurs indéfinis, la logique 4-valuée Belnap [5], représentée dans la Figure 1(c), a été introduite pour raisonner au sujet des bases de données inconsistantes, la logique 4-valuée de la Figure 1(d) [35] peut être utilisée pour modeler le désaccord entre deux sources de connaissance, l'ordre linéaire de la Figure 1(e) peut modeler différents niveaux d'incertitude et le treillis infini de la Figure 1(f) modèle l'ensemble de valeurs de vérité utilisé dans la logique fuzzy [69] (les réels entre 0 et 1).

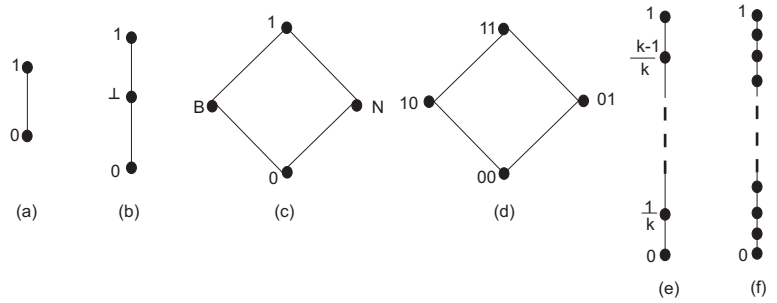

FIG. 1 – *Exemples de treillis de valeurs de vérité*

Un cas particulier d'algèbre de vérité est *l'algèbre quasi-booléenne* [83, 7, 34, 16] pour laquelle:

- $(B, \leq)$ est un treillis fini et distributif;
- (De Morgan) $\neg(a \wedge b) = \neg a \vee \neg b$ et $\neg(a \vee b) = \neg a \wedge \neg b$, pour tout a,$b \in B$;
- (involution) $\neg\neg a = a$, pour tout $a \in B$;
- (anti-monotonie) $a \leq b \Leftrightarrow \neg a \geq \neg b$, pour tout a,$b \in B$.

Par exemple, toutes les algèbres de la Figure 1 sont des algèbres quasi-booléennes.

Maintenant, on introduit le concept de structure logique, qui est convenable pour définir la logique de premier ordre multi-valuée. Si K est un ensemble non-vide d'éléments appelés *genres* (en anglais, kinds), une *signature logique* est une paire $(\mathcal{B},\Sigma_L)$, où $\mathcal{B}$ est une algèbre de vérité et Σ_L est un ensemble K^+-indexé d'ensembles distincts $\Sigma_L = (\Sigma_{L,w}|w \in K^+)$. Les éléments $w \in K^+$ sont appelés *des types logiques* et les éléments $p \in \Sigma_{L,w}$ sont appelés *des symboles prédicatifs du type* w.

Définition 2.2 Soit $(\mathcal{B},\Sigma_L)$ une signature logique. Une $(\mathcal{B},\Sigma_L)$-*structure logique* est une paire $\mathcal{S} = (S,\Sigma_L^{\mathcal{S}})$, où S est un ensemble K-indexé et $\Sigma_L^{\mathcal{S}}$ est un ensemble K^+-indexé d'interprétations de prédicats

$$\Sigma_L^{\mathcal{S}} = (\Sigma_{L,w}^{\mathcal{S}}|w \in K^+).$$

$\Sigma^{\mathcal{S}}_{L,w} = \{p^{\mathcal{S}} : S_w \to B | p \in \Sigma_{L,w}\}$ est un ensemble $\Sigma_{L,w}$-indexé de fonctions de S_w dans B.

La syntaxe de la logique de premier ordre sur une signature logique ($\mathcal{B}$,Σ_L) et sur un ensemble K-indexé de variables X est la suivante:

$$\phi = p(x_1, \ldots, x_m) \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid (\exists x)\phi \mid (\forall x)\phi,$$

pour tout prédicat p du type $k_1 \ldots k_m \in K^+$ et $x_i \in X_{k_i}$, $1 \leq i \leq m$.

On notera par $\mathcal{L}^O(\Sigma_L,X)$ l'ensemble de formules de la logique de premier ordre formées avec les prédicats de Σ_L, les variables de X et les connecteurs logiques de $O \subseteq \{\wedge, \vee, \neg\}$. La sémantique des formules est définie en utilisant les opérateurs $\neg$, $\wedge$ et $\vee$ de l'algèbre de vérité $\mathcal{B}$ et on notera par $[\phi]^{\mathcal{S}}$ la valeur de vérité de la formule $\phi \in \mathcal{L}^{\{\wedge,\vee,\neg\}}(\Sigma_L,X)$ dans la structure logique $\mathcal{S}$.

La logique temporelle CTL^* [24] décrit des séquences de transitions entre les états d'un système réactif qui réagit et répond de façon continue à son environnement. Elle est définie sur un ensemble de propositions atomiques AP et elle contient deux types de formules, *des formules d'état* et *des formules de chemin*. La syntaxe est donnée par les règles suivantes ($p \in AP$, φ est une formule d'état, et ψ est une formule de chemin):

- $true$, $false$ et p sont des formules d'état;
- si φ_1 et φ_2 sont des formules d'état, alors $\neg\varphi_1$, $\varphi_1 \vee \varphi_2$ et $\varphi_1 \wedge \varphi_2$ sont des formules d'état;
- si ψ est une formule de chemin, alors $\forall\psi$ et $\exists\psi$ sont des formules d'état;
- chaque formule d'état est une formule de chemin;
- si ψ_1 et ψ_2 sont des formules de chemin, alors $\neg\psi_1$, $\psi_1 \vee \psi_2$, $\psi_1 \wedge \psi_2$, $\mathbf{X}\psi_1$, $\overline{\mathbf{X}}\psi_1$, $\psi_1\,\mathbf{U}\psi_2$ et $\psi_1\,\mathbf{R}\psi_2$ sont des formules de chemin.

CTL^* est l'ensemble de toutes les formules d'état produites par les règles présentées ci-dessus; CTL^*_+ est le sous-ensemble de CTL^* qui consiste en formules sans négation, CTL est le sous-ensemble de CTL^* qui contient les formules dans lesquelles chaque opérateur de chemin est immédiatement précédé par un quantificateur de chemin, LTL est le sous-ensemble de CTL^* qui contient les formules de la forme $\forall\psi$, où ψ est une formule de chemin dans laquelle les seules sous-formules d'état autorisées sont les propositions atomiques et $\forall CTL^*$ ($\exists CTL^*$) est le sous-ensemble de CTL^* qui consiste en formules qui ne contiennent pas $\exists$ ($\forall$).

Une *structure Kripke multi-valuée* [43, 44] sur un ensemble de propositions atomiques AP et sur une algèbre de vérité $\mathcal{B} = (B, \wedge, \vee, \neg)$ est un uplet $M = (Q,R,L)$, où Q est un ensemble d'états, $R : Q \times Q \to B$ est un prédicat de transition multi-valuée et $L : Q \to (AP \to B)$ est une fonction d'étiquetage qui donne pour chaque état q les valeurs de vérité des propositions atomiques.

Soit D un sous-ensemble de B. Une *D-séquence infinie d'états* est une séquence $\pi = q_0 q_1 \ldots$ de sorte que $R(q_i,q_{i+1}) \in D$, pour tout $i \geq 0$. Une *D-séquence finie maximal d'états* est une séquence $\pi = q_0 q_1 \ldots q_n$ de sorte que $R(q_i,q_{i+1}) \in D$, pour tout $0 \leq i < n$, et $R(q_n,q) \notin D$, pour tout $q \in Q$. La longueur de π, notée par $|\pi|$, est ∞ dans le premier cas et $n+1$ dans le second. Un *D-chemin* de M est une D-séquence infinie d'états ou une D-séquence finie maximal d'états. $\Pi(M,D,q)$ représente l'ensemble de D-chemins de M qui commencent dans l'état q, $\Pi(M,D)$ représente l'ensemble de D-chemins de M et $\pi(i)$ représente l'état q_i.

La logique temporelle multi-valuée, mv-CTL^*, est interprétée sur des structures Kripke multi-valuées. On définit par extension de la logique temporelle 2-valuée, la valeur de vérité d'une CTL^*-formule d'état ϕ dans un état q d'une structure Kripke M, notée par $[\phi]_q^M$, et la valeur de vérité d'une CTL^*-formule de chemin ψ le long d'un chemin π d'une structure Kripke M, notée par $[\psi]_\pi^M$.

La logique temporelle de la connaissance $KCTL^*P$ [41] est convenable pour raisonner au sujet de la connaissance dans les *systèmes multi-agents*, i.e. les systèmes qui consistent dans une collection d'agents qui réagissent réciproquement. La modélisation de la connaissance est basée sur "les mondes possibles". L'intuition est que si un agent ne connaît pas complètement le monde, il considérera un nombre de mondes possibles (ceux-ci sont donnés par une *relation de similarité*). On dit que l'agent *connaît un fait* ϕ si ϕ est vrai dans toutes les mondes possibles de l'agent.

Soit AP un ensemble de propositions atomiques et n le nombre d'agents du système. Comme dans le cas de CTL^*, il y a deux types de formules, *des formules d'état* et *des formules de chemin*. Les règles qui décrivent sa syntaxe incluent celles de CTL^* avec les règles pour les opérateurs du temps passé [70] et les opérateurs de la connaissance [41]:

- si ψ_1 et ψ_2 sont des formules de chemin, alors $\mathbf{Pr}\ \psi_1$, $\psi_1\, \mathbf{S}\, \psi_2$ et $\psi_1\, \mathbf{B}\, \psi_2$ sont des formules de chemin;
- si φ est une formule d'état alors $\mathbf{K}_i\, \varphi$ et $\mathbf{P}_i\, \varphi$ sont des formules d'état, pour tout $1 \leq i \leq n$.

$KCTL^*P$ est l'ensemble de toutes les formules d'état produites par les règles présentées ci-dessus; $KCTL^*P_+$ est le sous-ensemble de $KCTL^*P$ qui consiste en formules sans négation et $\forall KCTL^*P$ ($\exists KCTL^*P$) est le sous-ensemble de $KCTL^*P$ qui consiste en formules qui ne contiennent pas $\exists$ et $\mathbf{P}_i$ ($\forall$ et $\mathbf{K}_i$).

La logique temporelle de la connaissance multi-valuée, mv-$KCTL^*P$, est interprétée sur des structures Kripke multi-agents multi-valuées.

Définition 2.3 Soit AP un ensemble de propositions atomiques et $\mathcal{B} =$

$(B, \wedge, \vee, \neg)$ une algèbre de vérité. Une *structure Kripke multi-agents multi-valuée* sur AP et $\mathcal{B}$ est un uplet $M = (Q,R,L,(\sim_i | \ 1 \leq i \leq n))$, où (Q,R,L) est une structure Kripke multi-valuée sur AP et $\mathcal{B}$ et $\sim_i$: $Q \times Q \rightarrow B$, pour tout $1 \leq i \leq n$, est la relation de similarité d'agent i. Chaque relation de similarité $\sim_i$ satisfait les propriétés suivantes:

- *réflexivité*: $\sim_i (x,x) = 1$, pour tout $x \in Q$;
- *symétrie*: $\sim_i (x,y) = \sim_i (y,x)$, pour tout $x,y \in Q$;
- *transitivité*: $\sim_i (x,z) = \sim_i (x,y) \wedge \sim_i (y,z)$, pour tout $x,y,z \in Q$.

Soit M une structure Kripke multi-agents multi-valuée sur AP et $\mathcal{B}$ et D un sous-ensemble de B. On définit l'ensemble de tous D-chemins de M, noté par $\Pi(M,D)$, exactement comme pour les structures Kripke multi-valuée. De plus, nous appellerons *un point*, toute paire (π,m), où $\pi \in \Pi(M,D)$ et $m \in \mathbf{N}$ avec $0 \leq m < |\pi|$. L'ensemble de points de la structure Kripke M sera noté par $\mathsf{Points}(M,D)$.

Pour n'importe quelle structure Kripke multi-agents, on peut obtenir un système interprété en déroulant le prédicat de transition. La fonction d'étiquetage sur les points est compatible avec la fonction d'étiquetage sur les états correspondants.

Définition 2.4 Soit $M = (Q,R,L,(\sim_i | \ 1 \leq i \leq n))$ une structure Kripke multi-agents multi-valuée. Le *système interprété* correspondant à M est un uplet $I = (\mathsf{Paths}(M,D),L^I,(\sim_i^I | \ 1 \leq i \leq n))$, où

- $L^I : \mathsf{Points}(M,D) \rightarrow (AP \rightarrow B)$ est une fonction d'interprétation pour les propositions atomiques définie par $L^I(\pi,m)(p) = L(\pi(m))(p)$, pour tout $\pi \in \mathsf{Paths}(M,D)$ et $0 \leq m < |\pi|$;
- $\sim_i^I$: $\mathsf{Points}(M,D) \times \mathsf{Points}(M,D) \rightarrow B$, pour tout $1 \leq i \leq n$, est la relation de similarité définie par $\sim_i^I ((\pi,m),(\pi',m')) = \sim_i (\pi(m),\pi'(m'))$.

À partir de maintenant nous ne ferons aucune distinction entre L et l'extension de L aux points L^I. La même chose est valable pour les relations de similarité. Si ϕ est une $KCTL^*P$-formule et $I = (\mathsf{Paths}(M,D),L,(\sim_i | \ 1 \leq i \leq n))$ est un système interprété, alors on définit *la valeur de vérité de* ϕ *dans le point* (π,m), notée par $[\phi]^I_{(\pi,m)}$, par extension de la logique temporelle de la connaissance 2-valuée.

3 Abstractions multi-valuées

Le deuxième chapitre de la thèse introduit les *abstractions multi-valuées*. Les abstractions sont obtenues en appliquant des relations d'équivalence et

après, les symboles prédicatifs de la logique sont redéfinis à s'appliquer correctement aux classes d'équivalence. Comme une classe d'équivalence peut contenir plus qu'un élément et chaque élément mène à une valeur de vérité pour chaque prédicat, redéfinir un prédicat sur une classe d'équivalence implique la définition d'une politique de recombinaison de valeurs de vérité d'un ensemble donné. Une telle politique est appelée une *politique d'interprétation* [40].

Définition 3.1 Soit $\mathcal{B} = (B, \wedge, \vee, \neg)$ une algèbre de vérité. Une politique d'interprétation sur $\mathcal{B}$ est une fonction α de B dans l'ensemble $\{\exists^S, \exists_a^{S'} | S, S' \in \mathcal{P}(B) - \{\emptyset\}\}$.

Une politique d'interprétation α sur $\mathcal{B}$ fonctionne de la manière suivante. Soit A un ensemble non-vide d'éléments, p un symbole prédicatif unaire et $\mathcal{I}_p^A : A \to B$ une fonction d'interprétation qui donne les valeurs de vérité de p pour chaque $a \in A$. Étant donné un ensemble arbitraire et non-vide $X \subseteq \mathcal{P}(A) - \{\emptyset\}$, on veut utiliser α pour définir une nouvelle fonction d'interprétation $\mathcal{I}_p^X : X \to B$. Pour chaque $T \in X$, $\mathcal{I}_p^X(T)$ est la valeur de vérité $b \in B$ si l'une des propriétés suivantes est satisfaite:

– si $\alpha(b) = \exists^S$, alors

$$(\forall t \in T)(\mathcal{I}_p^A(t) \in S) \ \wedge \ (\exists t \in T)(\mathcal{I}_p^A(t) = b)$$

– si $\alpha(b) = \exists_a^S$, alors

$$(\forall t \in T)(\mathcal{I}_p^A(t) \in S) \ \wedge \ (\exists t_1, t_2 \in T)(\mathcal{I}_p^A(t_1) \leq b \leq \mathcal{I}_p^A(t_2)).$$

C'est assez clair qu'on ne peut obtenir aucune fonction $\mathcal{I}_p^X$. Il peut se passer qu'aucune politique $\alpha(b)$ ne puisse être appliquée à $T \in X$ ou que deux politiques distinctes $\alpha(b)$ et $\alpha(b')$ puissent être appliquées à $T \in X$. Quand α mène à une fonction unique $\mathcal{I}_p^X$ comme ci-dessus, alors $\mathcal{I}_p^X$ sera appelé *la réintérpretation de* $\mathcal{I}_p^A$ *sur* X *selon* α et tout élément $\mathcal{I}_p^X(T)$ sera appelé *la valeur de vérité de* p *sur* T *selon* α (elle est notée par p_α^T).

Par conséquent, une politique d'interprétation a l'intention de redéfinir des fonctions d'interprétation déjà existantes. On peut remarquer qu'une politique de la forme $\alpha(b) = \exists^{\{b\}}$ dit qu'un prédicat p est réinterprété à b sur un sous-ensemble non-vide T si tous les éléments de T évaluent le prédicat p à b. Quelquefois, nous noterons cela par $\alpha(b) = \forall$.

Dans [90], une technique d'abstraction a été proposée pour les types abstraits de données. Cette technique emploie une logique de premier ordre sous l'interprétation 3-valuée Kleene. L'algèbre de vérité est basée sur le treillis

$(B = \{0,\perp,1\}, \leq)$ dont $0 \leq \perp \leq 1$. Trois types d'abstraction ont été définis: abstractions $\forall\forall$, abstractions $\forall\exists$ et abstractions $\exists^{0,1}\forall$.

On peut voir facilement que les réintérpretations des prédicats dans ces abstractions sont déterminées par les politiques d'interprétation suivantes:

type d'abstraction [90]	politique d'interprétation		
$\forall\forall$	$\alpha(0) = \exists^{\{0\}}$	$\alpha(\perp) = \exists_a^{\{0,\perp,1\}}$	$\alpha(1) = \exists^{\{1\}}$
$\forall\exists$	$\alpha(0) = \exists^{\{0,\perp,1\}}$	$\alpha(\perp) = \exists^{\{\perp,1\}}$	$\alpha(1) = \exists^{\{1\}}$
$\exists^{0,1}\forall$	$\alpha(0) = \exists^{\{0\}}$	$\alpha(\perp) = \exists^{\{0,\perp,1\}}$	$\alpha(1) = \exists^{\{0,1\}}$

Les abstractions de structures logiques sont définies comme des paires formées par une relation d'équivalence et une politique d'interprétation. Les classes d'équivalence représentent des ensembles d'éléments qui sont traités comme un tout et la politique d'interprétation est utilisée pour obtenir les valeurs de vérité des prédicats sur les classes d'équivalence.

Définition 3.2 Soit $\mathcal{S} = (S,\Sigma_L^{\mathcal{S}})$ une $(\mathcal{B},\Sigma_L)$-structure logique, ρ une relation d'équivalence sur S et α une politique d'interprétation sur $\mathcal{B}$. Une *α-abstraction de $\mathcal{S}$ par ρ* est une $(\mathcal{B},\Sigma_L)$-structure logique $\mathcal{S}' = (S/\rho,\Sigma_L^{\mathcal{S}'})$ de sorte que:

- $S/\rho = (S_k/\rho_k | k \in K)$;
- $p^{\mathcal{S}'}([a_1]_{\rho_{k_1}},\ldots,[a_m]_{\rho_{k_m}})$ est la valeur de p sur l'ensemble

$$\{(u_1,\ldots,u_m) | u_i \in [a_i]_{\rho_{k_i}}, 1 \leq i \leq m\}$$

  selon α, pour tout prédicat p du type $k_1 \ldots k_m$ avec $m \geq 1$, et $a_i \in S_{k_i}$, $1 \leq i \leq m$.

Pour être utile, une abstraction devrait préserver un ensemble spécifique de propriétés. Nous allons fournir trois types de résultats de préservation:

- *la $\geq$-préservation relative à un ensemble de propriétés P et deux valeurs de vérité b,$b' \in B$* qui signifie qu'une propriété donnée $\phi \in P$ est évaluée à une valeur de vérité plus grande que ou égale à b' dans le système concret, toutes les fois qu'elle est évaluée à une valeur de vérité plus grande que ou égale à b dans le système abstrait;
- *la $\leq$-préservation relative à un ensemble de propriétés P et deux valeurs de vérité b,$b' \in B$* qui signifie qu'une propriété donnée $\phi \in P$ est évaluée à une valeur de vérité moins que ou égale à b' dans le système concret, toutes les fois qu'elle est évaluée à une valeur de vérité moins que ou égale à b dans le système abstrait;

– *la =-préservation relative à un ensemble de propriétés P et un ensemble de valeurs de vérité B* qui signifie qu'une propriété donnée $\phi \in P$ est évaluée à une valeur de vérité $b \in B$ dans le système concret, toutes les fois qu'elle est évaluée à b dans le système abstrait.

La théorème suivante contient deux résultats de $\geq$-préservation et deux résultats de $\leq$-préservation. On utilise les notations $\downarrow b = \{y \in B \mid y \leq b\}$ et $\uparrow b = \{y \in B \mid y \geq b\}$, pour tout élément b d'un treillis B.

Théorème 3.1 Soit $\mathcal{S} = (S,\Sigma_L^{\mathcal{S}})$ une $(\mathcal{B},\Sigma_L)$-structure logique, ρ une équivalence sur S, α une politique d'interprétation sur $\mathcal{B}$, $\mathcal{S}' = (S/\rho,\Sigma_L^{\mathcal{S}'})$ une α-abstraction de $\mathcal{S}$ par ρ et b une valeur de vérité dans B.

1. S'il y a une valeur $b' \in B$ de sorte que $\alpha(x) \in \{\exists^T,\exists_a^T,\forall \mid T \subseteq\uparrow b'\}$, pour tout $x \geq b$, alors:

$$[\phi]^{\mathcal{S}'} \geq b \;\Rightarrow\; [\phi]^{\mathcal{S}} \geq b',$$

pour tout $\phi \in \mathcal{L}^{\{\wedge\}}(\Sigma_L,X)$. De plus, si

$$\vee B' \geq b \;\Rightarrow\; (\exists x \in B')(x \geq b), \text{ pour tout } B' \subseteq B,$$

alors

$$[\phi]^{\mathcal{S}'} \geq b \;\Rightarrow\; [\phi]^{\mathcal{S}} \geq b',$$

pour tout $\phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Sigma_L,X)$.

2. S'il y a $b' \in B$ de sorte que $\alpha(x) \in \{\exists^{S'},\exists_a^{S'},\forall \mid S' \subseteq\downarrow b'\}$, pour tout $x \leq b$, alors:

$$[\phi]^{\mathcal{S}'} \leq b \;\Rightarrow\; [\phi]^{\mathcal{S}} \leq b',$$

pour tout $\phi \in \mathcal{L}^{\{\vee\}}(\Sigma_L,X)$. De plus, si

$$\wedge B' \leq b \;\Rightarrow\; (\exists x \in B')(x \leq b), \text{ pour tout } B' \subseteq B,$$

alors

$$[\phi]^{\mathcal{S}'} \leq b \;\Rightarrow\; [\phi]^{\mathcal{S}} \leq b',$$

pour tout $\phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Sigma_L,X)$.

3. Si $\alpha(b) = \forall$ et $\alpha(x) \in \{\exists^S,\forall \mid S \subseteq\uparrow b \cap \downarrow x\}$, pour tout $x > b$, alors:

$$[\phi]^{\mathcal{S}'} \geq b \;\Rightarrow\; b \leq [\phi]^{\mathcal{S}} \leq [\phi]^{\mathcal{S}'} \text{ et}$$
$$[\phi]^{\mathcal{S}'} = b \;\Rightarrow\; [\phi]^{\mathcal{S}} = b,$$

pour tout $\phi \in \mathcal{L}^{\{\wedge\}}(\Sigma_L,X)$.

4. Si $\alpha(b) = \forall$ et $\alpha(x) \in \{\exists^S, \forall \mid S \subseteq\uparrow x \cap \downarrow b\}$, pour tout $x < b$, alors:

$$[\phi]^{\mathcal{S}'} \leq b \;\Rightarrow\; [\phi]^{\mathcal{S}'} \leq [\phi]^{\mathcal{S}} \leq b \text{ et}$$
$$[\phi]^{\mathcal{S}'} = b \;\Rightarrow\; [\phi]^{\mathcal{S}} = b,$$

pour tout $\phi \in \mathcal{L}^{\{\vee\}}(\Sigma_L, X)$.

Les résultats de =-préservation sont plus complexes comme l'on peut voir dans les conditions de la théorème suivante.

Théorème 3.2 Soit $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ une $(\mathcal{B}, \Sigma_L)$-structure logique, ρ une équivalence sur S, α une politique d'interprétation sur $\mathcal{B}$, $\mathcal{S}' = (S/\rho, \Sigma_L^{\mathcal{S}'})$ une α-abstraction de $\mathcal{S}$ par ρ et b une valeur de vérité dans B. Si:

1. $\alpha(b) = \forall$;
2. $\alpha(x) \in \{\exists^{S'}, \exists_a^{S'}, \forall \mid S' \subseteq\uparrow b\}$, pour tout $x > b$;
3. $\alpha(x) \in \{\exists^{S'}, \exists_a^{S'}, \forall \mid S' \subseteq\downarrow b\}$, pour tout $x < b$;
4. pour tout $B' \subseteq B$, $\wedge B' \leq b$ implique qu'il y a un élément $x \in B'$ de sorte que $x \leq b$;
5. pour tout $B' \subseteq B$, $\vee B' \geq b$ implique qu'il y a un élément $x \in B'$ de sorte que $x \geq b$;
6. pour tout $B' \subseteq B$, $\wedge B' = b$ implique que $b \in B'$;
7. pour tout $B' \subseteq B$, $\vee B' = b$ implique que $b \in B'$;

alors

$$[\phi]^{\mathcal{S}'} \geq b \;\Rightarrow\; [\phi]^{\mathcal{S}} \geq b,$$
$$[\phi]^{\mathcal{S}'} \leq b \;\Rightarrow\; [\phi]^{\mathcal{S}} \leq b,$$
$$[\phi]^{\mathcal{S}'} = b \;\Rightarrow\; [\phi]^{\mathcal{S}} = b,$$

pour tout $\phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Sigma_L, X)$.

Maintenant, nous montrons que les abstractions multi-valuées peuvent être calculées en utilisant seulement des abstractions 2-valuées, pourvu que quelques conditions soient remplies. Pour simplifier la présentation, nous considérerons seulement des abstractions 2-valuées basées sur les politiques d'interprétation sur $\mathcal{B}_2$ suivantes ($\mathcal{B}_2$ est l'algèbre de vérité avec deux éléments):

- $\alpha(0) = \exists$ et $\alpha(1) = \forall$;
- $\alpha'(0) = \forall$ et $\alpha'(1) = \exists$.

On peut remarquer que α correspond à l'interprétation par *sous-approximation* (i.e. $p([a]_\rho) = 1$ ssi $p(a_1) = 1$, pour tout $a_1 \in [a]_\rho$) et α' correspond

à l'interprétation par *sur-approximation* (i.e. $p([a]_\rho) = 1$ ssi $p(a_1) = 1$, pour un élément $a_1 \in [a]_\rho$).

Définition 3.3 Soit $\mathcal{S} = (S,\Sigma_L^{\mathcal{S}})$ une $(\mathcal{B}_2,\Sigma_L)$-structure logique et ρ une équivalence sur S. Une $\mathcal{P}$-*abstraction de* $\mathcal{S}$ *par* ρ, où $\mathcal{P} \subseteq \Sigma_L$, est une $(\mathcal{B}_2,\Sigma_L)$-structure logique $\mathcal{S}' = (S/\rho,\Sigma_L^{\mathcal{S}'})$ de sorte que:

- $S/\rho = (S_k/\rho_k | k \in K)$;
- $p^{\mathcal{S}'}([a_1]_{\rho_{k_1}},\ldots,[a_m]_{\rho_{k_m}})$ est la valeur de p sur l'ensemble

$$\{(u_1,\ldots,u_m) | u_i \in [a_i]_{\rho_{k_i}}, 1 \leq i \leq m\}$$

  selon α si $p \in \mathcal{P}$ et selon α' si $p \notin \mathcal{P}$, pour tout prédicat p du type $k_1 \ldots k_m$ avec $m \geq 1$, et $a_i \in S_{k_i}$, $1 \leq i \leq m$.

Le passage des abstractions 2-valuées aux abstractions multi-valuées suit deux pas:

- nous représentons les prédicats multi-valués par des ensembles de prédicats 2-valués en utilisant un nouveau ordre partiel sur B $\leq'$. Avec cette représentation, on associe une $(\mathcal{B}_2,\Sigma'_L)$-structure logique $\mathcal{S}_{\leq'}$ à toute $(\mathcal{B},\Sigma_L)$-structure logique $\mathcal{S}$;
- si $\mathcal{S}'_{\leq'}$ est une $\mathcal{P}$-abstraction de $\mathcal{S}_{\leq'}$ par une équivalence ρ, alors nous extrayons une politique d'interprétation α sur $\mathcal{B}$ de sorte que la $(\mathcal{B},\Sigma_L)$-structure logique $\mathcal{S}'$ associée à $\mathcal{S}'_{\leq'}$ est une α-abstraction de $\mathcal{S}$ par ρ.

Les relations qu'on vient de décrire ci-dessus nous permettent d'obtenir les abstractions multi-valuées en utilisant les procédures connues qui calculent des abstractions 2-valuées classiques.

Avant de discuter les abstractions multi-valuées de structures Kripke, nous présentons une étude de cas pour utiliser l'abstraction dans le contexte des modèles du contrôle d'accès [37].

Le contrôle d'accès est une des facettes de la réalisation des politiques de sécurité. Dans les modèles du contrôle d'accès, la politique de sécurité est rendue effective par l'attribution des privilèges d'accès aux objets qui composent le système et par les règles qui autorisent la création et/ou la destruction de nouveaux objets et la modification de leurs privilèges d'accès.

Un modèle puissant des systèmes du contrôle d'accès est *le modèle de la matrice d'accès* [54]. Dans ce modèle, *l'état de protection* du système est caractérisé par l'ensemble de privilèges d'accès que différentes entités (*sujets* ou *objets*) ont sur d'autres entités et par l'ensemble de *commandes* qui peuvent changer cet état par la création/destruction de sujets/objets ou par l'addition/effacement de privilèges d'accès. La force expressive de ce modèle est suffisamment grande pour inclure d'autres modèles comme les systèmes

"take-grant" [67], les systèmes SPM [88], les systèmes ESPM [1], les systèmes TAM [87], etc.

Le plus important problème de décision dans le modèle de la matrice d'accès est *le problème de sûreté*: étant donné deux entités A et B et un privilège R, il faut décider si le système peut évoluer dans un état dans lequel A a le privilège R sur B. Très tôt, on a montré que ce problème est non décidable [54] et il reste tel même pour les systèmes sans destruction de sujets ou objets [53]. Par conséquent, plusieurs restrictions ont été proposées [54, 67, 87] pour lesquelles le problème de sûreté est décidable.

On propose deux notions de simulation entre les systèmes de protection et on définit une classe de modèles du contrôle d'accès qui sont simulés par des modèles du contrôle d'accès avec un nombre fini d'objets (pour ces systèmes le problème de la sûreté est décidable). Alors, nous montrons que plusieurs classes de systèmes de protection de la littérature chutent dans cette classe, particulièrement *les systèmes "take-grant"* et *les systèmes de la matrice d'accès typée, monotones, avec un graphe de création acyclique*. Par ceci on unifie et clarifie la preuve de décidabilité du problème de sûreté pour ces classes de systèmes de protection.

On continue avec la présentation des abstractions multi-valuées des structures Kripke qui préservent des formules dans la logique temporelle multi-valuée mv-CTL^*.

Une abstraction d'une structure Kripke multi-valuée est obtenue en appliquant des relations d'équivalence sur l'espace d'états du système concret. Alors, les symboles prédicatifs de la logique sont redéfinis pour opérer correctement sur les classes d'équivalence en utilisant les politiques d'interprétation.

Définition 3.4 Soit $M = (Q,R,L)$ une structure Kripke multi-valuée sur AP et $\mathcal{B}$, ρ une équivalence sur Q, et α_R, α_L deux politiques d'interprétation sur $\mathcal{B}$. Une structure Kripke multi-valuée $M' = (Q',R',L')$ sur AP et $\mathcal{B}$ est appelée une (α_R,α_L)-*abstraction de* M *par* ρ si:

- $Q' = Q/_\rho$;
- R' est la réintérpretation de R sur $Q' \times Q'$ selon α_R;
- L' est la réintérpretation de L sur Q' selon α_L.

Les abstractions de structures Kripke introduites dans [39] sont des instances de la Définition 3.4.

Soit $M_2 = (Q_2,R_2,L_2)$ une (α_R,α_L)-abstraction d'une structure Kripke $M_1 = (Q_1,R_1,L_1)$ sur AP et $\mathcal{B} = (B, \leq)$ par une équivalence ρ et $D \subseteq B$. On dit qu'un chemin $\pi_2 \in \Pi(M_2,D)$ est *un chemin correspondant* à $\pi_1 \in \Pi(M_1,D)$ si:

- $|\pi_2| = |\pi_1|$;

– $\pi_2(i) = [\pi_1(i)]$, pour tout $0 \leq i < |\pi_2|$.

On notera par $C_{M_1}(\pi_2)$ l'ensemble de D-chemins de M_1 qui ont π_2 comme un D-chemin correspondant dans M_2. On continue par présenter les résultats de préservation qu'on a trouvé pour ces types d'abstraction.

Théorème 3.3 Soit $M_1 = (Q_1,R_1,L_1)$ une structure Kripke sur AP et $\mathcal{B} = (B, \leq)$, $M_2 = (Q_2,R_2,L_2)$ une (α_R,α_L)-abstraction de M_1 par une équivalence ρ, $D \subseteq B$, et ϕ (ψ) une $\forall CTL^*_+$-formule d'état (de chemin) sur AP. Si

1. pour tout $\pi_1 \in \Pi(M_1,D)$ il y a un chemin correspondant $\pi_2 \in \Pi(M_2,D)$;
2. pour tout $q \in Q_1$ et $p \in AP$, $L_2([q])(p) \in D$ implique $L_1(q)(p) \in D$;
3. $b' \in D$ toute les fois qu'il y a un élément $b \in D$ de sorte que $b \leq b'$;
4. D contient la plus grande borne inférieure de chaque sous-ensemble non-vide de D;
5. si D contient la plus petite borne supérieure d'un sous-ensemble non-vide X de B, alors il contient tous les éléments de X,

alors

$$(\forall q \in Q_1)([\phi]^{M_2}_{[q]} \in D \quad \Rightarrow \quad [\phi]^{M_1}_q \in D)$$

et

$$(\forall \pi_2 \in \Pi(M_2,D))([\psi]^{M_2}_{\pi_2} \in D \quad \Rightarrow \quad (\forall \pi_1 \in C_{M_1}(\pi_2))([\psi]^{M_1}_{\pi_1} \in D)).$$

Remarque 3.1 Les premières deux conditions dans la Théorème 3.3 peuvent être prouvées en regardant seulement les politiques d'interprétation utilisées dans l'abstraction. Ainsi, la première condition est remplie si

$$(\alpha_R(b) = \exists^S \;\Rightarrow\; S \cap D = \emptyset) \wedge (\alpha_R(b) = \exists^S_a \;\Rightarrow\; S \cap D = b\downarrow \cap D = b\uparrow \cap D = \emptyset),$$

pour tout $b \in B - D$, pendant que la seconde est remplie si

$$(\alpha_L(d) = \exists^S \;\Rightarrow\; S \subseteq D) \;\wedge\; (\alpha_L(d) = \exists^S_a \;\Rightarrow\; S \cup d\downarrow \cup d\uparrow \subseteq D),$$

pour tout $d \in D$.

Théorème 3.4 Soit $M_1 = (Q_1,R_1,L_1)$ une structure Kripke sur AP et $\mathcal{B} = (B, \leq)$, $M_2 = (Q_2,R_2,L_2)$ une (α_R,α_L)-abstraction de M_1 par une équivalence ρ, $D \subseteq B$, $b \in B$ et ϕ (ψ) une $\forall CTL^*_+$-formule d'état (de chemin) sur AP. Si

1. pour tout $\pi_1 \in \Pi(M_1,D)$ il y a un chemin correspondant $\pi_2 \in \Pi(M_2,D)$;
2. pour tout $q \in Q_1$ et $p \in AP$, $L_2([q])(p) \geq b$ implique $L_1(q)(p) \geq b$;
3. pour tout $q,q' \in Q_1$, $R_2([q],[q']) \geq b$ implique $R_1(q,q') \geq b$;

4. pour tout sous-ensemble B' de B, $\vee B' \geq b$ implique $b' \geq b$ pour un $b' \in B'$,

alors:

$$(\forall q \in Q_1)([\phi]^{M_2}_{[q]} \geq b \quad \Rightarrow \quad [\phi]^{M_1}_{q} \geq b)$$

et

$$(\forall \pi_2 \in \Pi(M_2,D))([\psi]^{M_2}_{\pi_2} \geq b \quad \Rightarrow \quad (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{M_1}_{\pi_1} \geq b)).$$

Remarque 3.2 Tout comme c'était le cas avec la théorème antérieure, les premières trois conditions dans la Théorème 3.4 peuvent être prouvées en regardant seulement les politiques d'interprétation utilisées dans l'abstraction. Ainsi, la première condition est remplie si

$$(\alpha_R(b) = \exists^S \;\Rightarrow\; S \cap D = \emptyset) \wedge (\alpha_R(b) = \exists^S_a \;\Rightarrow\; S \cap D = b\downarrow \cap D = b\uparrow \cap D = \emptyset),$$

pour tout $b \in B - D$, pendant que la seconde et la troisième sont remplies si

$$(\alpha(d) = \exists^S \;\Rightarrow\; S \subseteq b\uparrow) \;\wedge\; (\alpha(d) = \exists^S_a \;\Rightarrow\; S \cup d\downarrow \cup d\uparrow \subseteq b\uparrow),$$

pour tout $\alpha \in \{\alpha_R, \alpha_L\}$ et $d \geq b$.

Théorème 3.5 Soit $M_1 = (Q_1, R_1, L_1)$ une structure Kripke sur AP et $\mathcal{B} = (B, \leq)$, $M_2 = (Q_2, R_2, L_2)$ une (α_R, α_L)-abstraction de M_1 par une équivalence ρ, $D \subseteq B$, $b \in B$ et ϕ (ψ) une $\forall CTL^*_+$-formule d'état (de chemin) sur AP. Si

1. pour tout $\pi_2 \in \Pi(M_2, D)$ il y a un D-chemin $\pi_1 \in C_{M_1}(\pi_2)$;
2. pour tout $q \in Q_1$ et $p \in AP$, $L_2([q])(p) \leq b$ implique $L_1(q)(p) \leq b$;
3. pour tout $q, q' \in Q_1$, $R_2([q],[q']) \leq b$ implique $R_1(q,q') \leq b$;
4. pour tout sous-ensemble B' de B, $\wedge B' \leq b$ implique $b' \leq b$ pour un $b' \in B'$,

alors:

$$(\forall q \in Q_1)([\phi]^{M_2}_{[q]} \leq b \quad \Rightarrow \quad [\phi]^{M_1}_{q} \leq b)$$

et

$$(\forall \pi_2 \in \Pi(M_2,D))([\psi]^{M_2}_{\pi_2} \leq b \quad \Rightarrow \quad (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{M_1}_{\pi_1} \leq b)).$$

Remarque 3.3 De même, la première condition dans la Théorème 3.5 est remplie si

$$(\alpha_R(b) = \exists^S \;\Rightarrow\; S \subseteq D) \;\wedge\; (\alpha_R(b) = \exists^S_a \;\Rightarrow\; S \cup b\downarrow \cup b\uparrow \subseteq D),$$

pour tout $b \in D$, pendant que la seconde et la troisième sont remplies si

$$(\alpha(d) = \exists^S \;\Rightarrow\; S \subseteq b\downarrow) \;\wedge\; (\alpha(d) = \exists^S_a \;\Rightarrow\; S \cup d\downarrow \cup d\uparrow \subseteq b\downarrow),$$

pour tout $\alpha \in \{\alpha_R, \alpha_L\}$ et $d \leq b$.

Théorème 3.6 Soit $M_1 = (Q_1,R_1,L_1)$ une structure Kripke sur AP et $\mathcal{B} = (B, \leq)$, $M_2 = (Q_2,R_2,L_2)$ une (α_R,α_L)-abstraction de M_1 par une équivalence ρ, $D \subseteq B$, $b \in B$ et ϕ (ψ) une $\exists CTL^*_+$-formule d'état (de chemin) sur AP. Si

1. pour tout $\pi_2 \in \Pi(M_2,D)$ il y a un D-chemin $\pi_1 \in C_{M_1}(\pi_2)$;
2. pour tout $q \in Q_1$ et $p \in AP$, $L_2([q])(p) \geq b$ implique $L_1(q)(p) \geq b$;
3. pour tout $q,q' \in Q_1$, $R_2([q],[q']) \geq b$ implique $R_1(q,q') \geq b$;
4. pour tout sous-ensemble B' de B, $\vee B' \geq b$ implique $b' \geq b$ pour un $b' \in B'$,

alors:

$$(\forall q \in Q_1)([\phi]^{M_2}_{[q]} \geq b \quad \Rightarrow \quad [\phi]^{M_1}_q \geq b)$$

et

$$(\forall \pi_2 \in \Pi(M_2,D))([\psi]^{M_2}_{\pi_2} \geq b \quad \Rightarrow \quad (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{M_1}_{\pi_1} \geq b)).$$

Remarque 3.4 De même, la première condition dans la Théorème 3.6 est remplie si

$$(\alpha_R(b) = \exists^S \ \Rightarrow \ S \subseteq D) \ \wedge \ (\alpha_R(b) = \exists^S_a \ \Rightarrow \ S \cup b\downarrow \cup b\uparrow \subseteq D),$$

pour tout $b \in D$, pendant que la seconde et la troisième sont remplies si

$$(\alpha(d) = \exists^S \ \Rightarrow \ S \subseteq b\uparrow) \ \wedge \ (\alpha(d) = \exists^S_a \ \Rightarrow \ S \cup d\downarrow \cup d\uparrow \subseteq b\uparrow),$$

pour tout $\alpha \in \{\alpha_R,\alpha_L\}$ et $d \geq b$.

Théorème 3.7 Soit $M_1 = (Q_1,R_1,L_1)$ une structure Kripke sur AP et $\mathcal{B} = (B, \leq)$, $M_2 = (Q_2,R_2,L_2)$ une (α_R,α_L)-abstraction de M_1 par une équivalence ρ, $D \subseteq B$, $b \in B$ et ϕ (ψ) une $\exists CTL^*_+$-formule d'état (de chemin) sur AP. Si

1. pour tout $\pi_1 \in \Pi(M_1,D)$ il y a un chemin correspondant $\pi_2 \in \Pi(M_2,D)$;
2. pour tout $q \in Q_1$ et $p \in AP$, $L_2([q])(p) \leq b$ implique $L_1(q)(p) \leq b$;
3. pour tout $q,q' \in Q_1$, $R_2([q],[q']) \leq b$ implique $R_1(q,q') \leq b$;
4. pour tout sous-ensemble B' de B, $\wedge B' \leq b$ implique $b' \leq b$ pour un $b' \in B'$,

alors:

$$(\forall q \in Q_1)([\phi]^{M_2}_{[q]} \leq b \quad \Rightarrow \quad [\phi]^{M_1}_q \leq b)$$

et

$$(\forall \pi_2 \in \Pi(M_2,D))([\psi]^{M_2}_{\pi_2} \leq b \quad \Rightarrow \quad (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{M_1}_{\pi_1} \leq b)).$$

Remarque 3.5 De même, la première condition dans la Théorème 3.7 est remplie si

$$(\alpha_R(b) = \exists^S \;\Rightarrow\; S \cap D = \emptyset) \wedge (\alpha_R(b) = \exists_a^S \;\Rightarrow\; S \cap D = b\downarrow \cap D = b\uparrow \cap D = \emptyset),$$

pour tout $b \in B - D$, pendant que la seconde et la troisième sont remplies si

$$(\alpha(d) = \exists^S \;\Rightarrow\; S \subseteq b\downarrow) \;\wedge\; (\alpha(d) = \exists_a^S \;\Rightarrow\; S \cup d\downarrow \cup d\uparrow \subseteq b\downarrow),$$

pour tout $\alpha \in \{\alpha_R, \alpha_L\}$ et $d \leq b$.

Remarque 3.6 Soit $M_1 = (Q_1, R_1, L_1)$ une structure Kripke sur AP et $\mathcal{B} = (B, \leq)$, $M_2 = (Q_2, R_2, L_2)$ une (α_R, α_L)-abstraction de M_1 par une équivalence ρ, $D \subseteq B$ et $b \in B$. Si toutes les conditions des Théorèmes 3.4, 3.5, 3.6 et 3.7 sont remplies et, de plus,

- pour tout $q' \in Q_1$ et $p \in AP$, $L_2([q'])(p) = b$ implique $L_1(q')(p) = b$;
- pour tout $q', q'' \in Q_1$, $R_2([q'], [q'']) = b$ implique $R_1(q', q'') = b$;
- pour tout sous-ensemble B' de B, $\wedge B' = b$ implique $b \in B'$;
- pour tout sous-ensemble B' de B, $\vee B' = b$ implique $b \in B'$;

alors,

$$(\forall q \in Q_1)([\phi]_{[q]}^{M_2} = b \;\Rightarrow\; [\phi]_q^{M_1} = b) \text{ et}$$

$$(\forall \pi_2 \in \Pi(M_2, D))([\psi]_{\pi_2}^{M_2} = b \;\Rightarrow\; (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]_q^{M_1} = b)),$$

pour toute ϕ (ψ) une CTL_+^*-formule d'état (de chemin) sur AP.

Quand on veut vérifier un système par abstraction, nous choisissons d'abord une équivalence et après, un type d'abstraction (une paire de politiques d'interprétation (α_R, α_L)). Il est possible que l'abstraction ne nous permette pas de tirer une conclusion (à l'égard de la propriété qu'on veut vérifier) et, par conséquent, on peut essayer de changer l'abstraction: l'équivalence ou le type d'abstraction ou les deux. Le choix le plus simple est de changer le type d'abstraction. Dans un tel cas, c'est utile de savoir les rapports entre les types d'abstraction pour éviter les types qui nous mèneraient aux mêmes conclusions.

Les Figures 2 et 3 contiennent les rapports entre les types d'abstraction des structures Kripke obtenues en utilisant les politiques d'interprétation sur $\mathcal{B}_3$ suivantes [39] ($\mathcal{B}_3$ est l'algèbre de vérité qui correspond a l'interprétation 3-valuée Kleene) :

- $\alpha_1(0) = \forall$, $\alpha_1(\perp) = \exists_a^{\{0,\perp,1\}}$ et $\alpha_1(1) = \forall$;
- $\alpha_2(0) = \exists^{\{0,\perp,1\}}$, $\alpha_2(\perp) = \exists^{\{\perp,1\}}$ et $\alpha_2(1) = \forall$;
- $\alpha_3(0) = \forall$, $\alpha_3(\perp) = \exists^{\{0,\perp\}}$ et $\alpha_3(1) = \exists^{\{0,\perp,1\}}$.

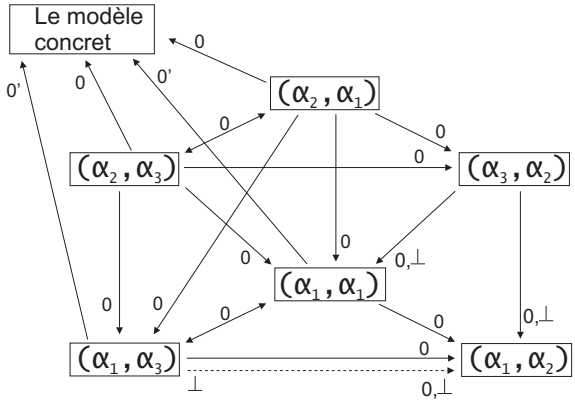

FIG. 2 – *Préserver les valeurs de vérité 0 dans les types d'abstraction possibles.*

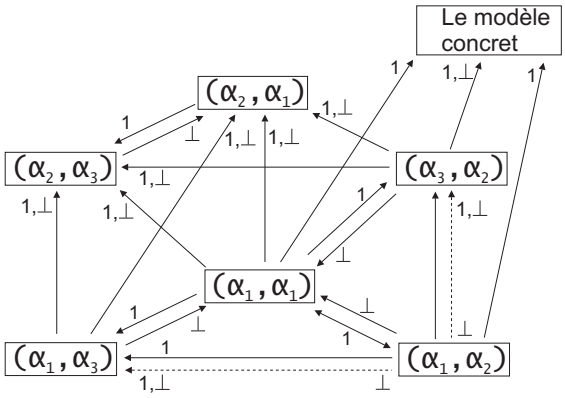

FIG. 3 – *Préserver les valeurs de vérité 1 et ⊥ dans les types d'abstraction possibles.*

Les deux figures montrent des résultats de préservation des formules de LTL_+: la Figure 2 concerne la valeur de vérité 0 pendant que l'autre concerne les valeurs de vérité 1 et $\perp$. Par exemple, une flèche de (α_2,α_3) à (α_1,α_1) étiquetée par 0 signifie qu'une propriété qui est évaluée à 0 dans une (α_2,α_3)-abstraction est aussi évaluée à 0 dans une (α_1,α_1)-abstraction (les deux abstractions sont sous la même équivalence). Une flèche comme celle de (α_1,α_2) à (α_1,α_3) étiquetée par $\perp$ et 1,$\perp$ (voir la Figure 3) signifie qu'une propriété qui est évaluée à $\perp$ dans une (α_1,α_2)-abstraction est évaluée à 1 ou $\perp$ dans une (α_1,α_3)-abstraction. Les flèches étiquetées par $0'$ représentent des résultats de préservation de la valeur 0 seulement pour les formules LTL_+ qui ne contiennent pas l'opérateur $\overline{\mathbf{X}}$.

La technique d'abstraction pour les structures Kripke multi-valuées peut être étendue même pour les structures Kripke multi-agents [38]. Comme on a déjà vu, le prédicat de transition et la fonction d'étiquetage du système abstrait sont obtenues par la réintérpretation selon une politique d'interprétation. Dans le cas des relations de similarité du système abstrait, nous ne pouvons utiliser toujours les réintérpretations des relations de similarité du système concret selon une politique d'interprétation. Il est possible que ces réintérpretations ne satisfassent pas la réflexivité, la symétrie ou la transitivité. Nous discuterons les réintérpretations utilisées dans [38] qui correspondent aux politiques d'interprétation sur $\mathcal{B}_3$ α_1, α_2 et α_3.

Les relations entre les réintérpretations selon ces politiques d'interprétation et les propriétés d'une relation de similarité sont énumérées dans la prochaine proposition.

Proposition 3.1 Étant donné $M = (Q,R,L,(\sim_i|\ 1 \leq i \leq n))$ une structure Kripke multi-agents multi-valuée sur AP et $\mathcal{B}_3$ et ρ une équivalence sur Q,

1. la réintérpretation de $\sim_i$ selon α_1 est symétrique;
2. la réintérpretation de $\sim_i$ selon α_2 est symétrique et transitive;
3. la réintérpretation de $\sim_i$ selon α_3 est réflexive et symétrique;

La proposition ci-dessus mentionne toutes les propriétés d'une relation de similarité qui sont toujours satisfaites par une réintérpretation de $\sim_i$ selon α_1, α_2 ou α_3. Par conséquent, quand on essaie de redéfinir les relations de similarité dans le système abstrait, il faut appliquer des clôtures réflexives ou transitives.

Définition 3.5 Soit $M = (Q,R,L,(\sim_i|\ 1 \leq i \leq n))$ une structure Kripke multi-agents multi-valuée sur AP et $\mathcal{B}_3$, ρ une équivalence sur Q, et α_R, α_L, $\alpha_S \in \{\alpha_1,\alpha_2,\alpha_3\}$ trois politiques d'interprétation sur $\mathcal{B}_3$. Une structure Kripke multi-agents $M' = (Q',R',L',(\sim'_i|\ 1 \leq i \leq n))$ sur AP et $\mathcal{B}_3$ est appelée *une* $(\alpha_R,\alpha_L,\alpha_S)$*-abstraction de* M *par* ρ si:

- $Q' = Q/_\rho$;
- R' est la réintérpretation de R sur $Q' \times Q'$ selon α_R;
- L' est la réintérpretation de L sur Q' selon α_L;
- $\sim'_i$ est défini de la façon suivante:
  - si $\alpha_S = \alpha_1$ alors $\sim'_i$ est la clôture réflexive et transitive de la réintérpretation de $\sim_i$ sur $Q' \times Q'$ selon α_1;
  - si $\alpha_S = \alpha_2$ alors $\sim'_i$ est la clôture réflexive de la réintérpretation de $\sim_i$ sur $Q' \times Q'$ selon α_2;
  - si $\alpha_S = \alpha_3$ alors $\sim'_i$ est la clôture transitive de la réintérpretation de $\sim_i$ sur $Q' \times Q'$ selon α_3.

Pour cette technique d'abstraction, on offre trois formes de préservation de propriétés, les deux premières étant fréquemment trouvées dans la littérature [31]:

- *la préservation faible relative à un ensemble de propriétés* P: si une propriété $\phi \in P$ est évaluée à 1 dans le système abstrait, alors ϕ est évaluée à 1 dans le système concret;
- *la préservation d'erreurs relative à un ensemble de propriétés* P: si une propriété $\phi \in P$ est évaluée à 0 dans le système abstrait, alors ϕ est évaluée à 0 dans le système concret;
- *la préservation très faible relative à un ensemble de propriétés* P: si une propriété $\phi \in P$ est évaluée à 1 ou $\perp$ dans le système abstrait, alors ϕ est évaluée à 1 ou $\perp$ dans le système concret;

Par la suite, on va présenter des résultats de préservation faibles ou très faibles.

Théorème 3.8 Soit $M_1 = (Q_1,R_1,L_1,(\sim_i^1 | \ 1 \leq i \leq n))$ une structure Kripke multi-agents sur AP et $\mathcal{B}_3$, ρ une équivalence sur Q_1, $M_2 = (Q_2,R_2,L_2,(\sim_i^2 | \ 1 \leq i \leq n))$ une $(\alpha_3,\alpha_L,\alpha_3)$-abstraction de M_1 par ρ et ϕ une $\forall KCTL^*P_+$-formule.

- Si $\alpha_L = \alpha_2$ alors

$$([\phi]^{I_2}_{(\pi_2,m)} \in \{\perp,1\}) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} \in \{\perp,1\}),$$

  pour tout $(\pi_2,m) \in \mathsf{Points}(M_2,\{\perp,1\})$. De plus, si $R_1(q,q') \in \{0,1\}$ et $\sim_i^1 (q,q') \in \{0,1\}$, pour tout $q,q' \in Q_1$ et $1 \leq i \leq n$ alors

$$([\phi]^{I_2}_{(\pi_2,m)} = 1) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 1),$$

  pour tout $(\pi_2,m) \in \mathsf{Points}(M_2,\{\perp,1\})$.
- Si $\alpha_L = \alpha_1$, $R_1(q,q') \in \{0,1\}$ et $\sim_i^1 (q,q') \in \{0,1\}$, pour tout $q,q' \in Q_1$ et $1 \leq i \leq n$ alors

$$([\phi]^{I_2}_{(\pi_2,m)} = 1) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 1),$$

  pour tout $(\pi_2,m) \in \mathsf{Points}(M_2,\{\perp,1\})$.

Théorème 3.9 Soit $M_1 = (Q_1,R_1,L_1,(\sim_i^1 | \ 1 \leq i \leq n))$ une structure Kripke multi-agents sur AP et $\mathcal{B}_3$, ρ une équivalence sur Q_1, $M_2 = (Q_2,R_2,L_2,(\sim_i^2 | \ 1 \leq i \leq n))$ une $(\alpha_1,\alpha_L,\alpha_1)$-abstraction de M_1 par ρ et ϕ une $\forall KCTL^*P_+$-formule.

- Si $\alpha_L \in \{\alpha_1,\alpha_2\}$ alors

$$([\phi]^{I_2}_{(\pi_2,m)} = 1) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 1),$$

  pour tout $(\pi_2,m) \in \mathsf{Points}(M_2,\{\perp,1\})$.

Théorème 3.10 Soit $M_1 = (Q_1,R_1,L_1,(\sim_i^1 \mid 1 \leq i \leq n))$ une structure Kripke multi-agents sur AP et $\mathcal{B}_3$, ρ une équivalence sur Q_1 pour laquelle $q \ \rho \ q'$ implique $\sim_i^1 (q,q') = 1$, $M_2 = (Q_2,R_2,L_2,(\sim_i^2 \mid 1 \leq i \leq n))$ une $(\alpha_2,\alpha_L,\alpha_2)$-abstraction de M_1 par ρ et ϕ une $\exists KCTL^*P_+$-formule.

– Si $\alpha_L \in \{\alpha_1,\alpha_2\}$ alors

$$([\phi]^{I_2}_{(\pi_2,m)} = 1) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 1),$$

pour tout $(\pi_2,m) \in \mathsf{Points}(M_2,\{\bot,1\})$.

– Si $\alpha_L = \alpha_2$ alors

$$([\phi]^{I_2}_{(\pi_2,m)} = \bot) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} \in \{\bot,1\}),$$

pour tout $(\pi_2,m) \in \mathsf{Points}(M_2,\{\bot,1\})$.

Maintenant, on continue avec la présentation des résultats de préservation d'erreurs.

Théorème 3.11 Soit $M_1 = (Q_1,R_1,L_1,(\sim_i^1 \mid 1 \leq i \leq n))$ une structure Kripke multi-agents sur AP et $\mathcal{B}_3$, ρ une relation d'équivalence sur Q_1, $M_2 = (Q_2,R_2,L_2,(\sim_i^2 \mid 1 \leq i \leq n))$ une $(\alpha_3,\alpha_L,\alpha_3)$-abstraction de M_1 par ρ, $\alpha_L \in \{\alpha_1,\alpha_3\}$ et ϕ une $\exists KCTL^*P_+$-formule. Alors

$$([\phi]^{I_2}_{(\pi_2,m)} = 0) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 0),$$

pour tout $(\pi_2,m) \in \mathsf{Points}(M_2,\{\bot,1\})$.

Théorème 3.12 Soit $M_1 = (Q_1,R_1,L_1,(\sim_i^1 \mid 1 \leq i \leq n))$ une structure Kripke multi-agents sur AP et $\mathcal{B}_3$, ρ une équivalence sur Q_1 pour laquelle $q \ \rho \ q'$ implique $\sim_i^1 (q,q') = 1$, $M_2 = (Q_2,R_2,L_2,(\sim_i^2 \mid 1 \leq i \leq n))$ une $(\alpha_2,\alpha_L,\alpha_2)$-abstraction de M_1 par ρ, $\alpha_L \in \{\alpha_1,\alpha_3\}$ et ϕ une $\forall KCTL^*P_+$-formule. Alors

$$([\phi]^{I_2}_{(\pi_2,m)} = 0) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 0),$$

pour tout $(\pi_2,m) \in \mathsf{Points}(M_2,\{\bot,1\})$.

Théorème 3.13 Soit $M_1 = (Q_1,R_1,L_1,(\sim_i^1 \mid 1 \leq i \leq n))$ une structure Kripke multi-agents sur AP et $\mathcal{B}_3$, ρ une relation d'équivalence sur Q_1, $M_2 = (Q_2,R_2,L_2,(\sim_i^2 \mid 1 \leq i \leq n))$ une $(\alpha_1,\alpha_L,\alpha_1)$-abstraction de M_1 par ρ, $\alpha_L \in \{\alpha_1,\alpha_3\}$ et ϕ une $\exists KCTL^*P_+$-formule. Alors

$$([\phi]^{I_2}_{(\pi_2,m)} = 0) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 0),$$

pour tout $(\pi_2,m) \in \mathsf{Points}(M_2,\{\bot,1\})$.

4 Abstractions de types de données

La majorité des techniques d'abstraction que nous avons trouvées dans la littérature sont conduites par presque le même mécanisme (par exemple, la fonction surjective) et elles reposent sur des résultats de préservation de propriétés semblables mais, les formalismes utilisés sont complètement différents. Nous pouvons dire que toutes ces techniques d'abstraction ont leurs racines dans l'interprétation abstraite de Cousot [30] mais, ce cadre offre seulement une méthodologie générale qui, dans des cas particuliers, soit complétée par des techniques spécifiques. Par conséquent, le développement de formalismes d'abstraction spécialisés permettant des réalisations raisonnables dans des cas pratiques est nécessaire.

Nous fournirons ci-dessous une solution à ce problème relatif aux types de données, qui élargit celle de [90]. On essaie de capturer l'essence de la réduction de types de données et pour cela, les types (abstraits) de données sont modelés par des algèbres d'appartenance [77] enrichies par des ensembles de symboles prédicatifs. Celles-ci représentent un formalisme largement accepté pour spécifier les types de données qui offre de la précision mathématique et peut être utilisé dans la pratique. Beaucoup de langages de programmation modernes, tels que C++ et Java, permettent aux utilisateurs de définir des types abstraits de données au-delà des types basiques. On définit une abstraction comme une paire formée d'une congruence et d'une politique d'interprétation. La congruence divise le type de données original et redéfinit ses opérations pour opérer correctement sur le type de données quotient. La politique d'interprétation interprète les symboles prédicatifs dans le type de données quotient. On montre que la politique d'interprétation ne peut être substituée par la congruence comme dans le cas des opérations. Par conséquent, l'abstraction du type de données devrait inclure nécessairement une politique d'interprétation.

Soit K un ensemble de genres. *Les signatures d'appartenance étendues logiquement* ajoutent des symboles prédicatifs aux signatures d'appartenance normales. Les symboles prédicatifs ont deux rôles:

- spécifier des propriétés de base satisfaites par les éléments d'une algèbre;
- construire des formules qui définissent de nouvelles propriétés.

Définition 4.1 Une *signature d'appartenance étendue logiquement* est un 4-uplet $\Omega_L = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$, où:

- $\mathcal{B} = (B, \wedge, \vee, \neg)$ est une algèbre de vérité;
- $\leq'$ est un ordre partiel sur B de sorte que:
  - $(B, \leq')$ est *un treillis inf-complet*, i.e. chaque sous-ensemble $B' \subseteq$

B a la plus grande borne inférieure. On note par $0'$ le plus petit élément de B.

  - pour tout $x \in B$, il y a un unique $y \in B$ de sorte que $x \succ' y$ ($<'=\leq' -\{(a,a)|a \in B\}$, $>'$ est l'inverse de $<'$ et $x \succ' y$ si $x >' y$ et il n'y a pas de $c \in B$ de sorte que $x >' c >' y$).

- $K' \subseteq K$ est un ensemble de *genres de base*
- $\Sigma_L = (\Sigma_{L,w}|w \in K'^+)$ est un ensemble de *symboles prédicatifs* (w est appelé un *type logique*)
- l'ensemble de genres K contient K' et un genre pour chaque type logique pour lequel nous avons au moins un symbole prédicatif dans Σ_L. Formellement,

$$K = K' \cup \{k_w|w \in K'^+ \text{ et } \Sigma_{L,w} \neq \emptyset\};$$

- Σ est un ensemble $K^* \times K$-indexé de symboles fonctionnels qui contient un symbole fonctionnel $(\_ , \cdots , \_)$ pour chaque type logique pour lequel nous avons au moins un symbole prédicatif dans Σ_L. Le symbole fonctionnel $(\_ , \cdots , \_)$ associé au type logique $w = k_1 \cdots k_m \in K'^+$ a l'arité $(k_1 \cdots k_m, k_w)$;
- l'ensemble de sortes S contient une sorte distinguée $s_{p,b}$ pour chaque $p \in \Sigma_L$ et $b \in B - \{0'\}$;
- π est une fonction de S dans K de sorte que $\pi(s_{p,b}) = k_w$, pour tout $p \in \Sigma_{L,w}$, $b \in B - \{0'\}$ et $w \in K'^+$.

Le genre k_w, pour un type logique $w = k_1 \ldots k_m \in K'^+$, représente tout les m-uplets pour lesquels l'élément sur la position i est de genre k_i (par le biais du symbole fonctionnel correspondant $(\_ , \cdots , \_)$) et $s_{p,b}$ définit l'ensemble de tous les nuplets sur lesquels p a des valeurs de vérité plus grandes que ou égales à b (par rapport à $\leq'$), pour tout $b \in B - \{0'\}$. Les nuplets sur lesquels p est $0'$ sont ceux qui n'appartiennent à aucune sorte $s_{p,b}$ avec $b \in B - \{0'\}$.

Étant donné une signature d'appartenance étendue logiquement $\Omega_L = (\mathcal{B}, \leq' , \Sigma, \Sigma_L, \pi)$ et un ensemble K-indexé de variables X, on définit l'ensemble de formules de premier ordre sur Ω_L et X de la manière suivante:

$$\phi = p(t_1, \ldots, t_m) \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid (\exists x)\phi \mid (\forall x)\phi,$$

pour tout prédicat p du type $k_1 \ldots k_m \in K^+$ et terme t_i de genre k_i, $1 \leq i \leq m$.

On notera par $\mathcal{L}^O(\Omega_L, X)$ l'ensemble de formules de premier ordre formées avec les prédicats de Ω_L, les variables de X et les connecteurs logiques de $O \subseteq \{\wedge, \vee , \neg\}$. La sémantique des formules est définie de la même manière

que pour les structures logiques et on notera par $[\phi]^{\mathcal{A}}$ la valeur de vérité de la formule $\phi \in \mathcal{L}^{\{\wedge,\vee,\neg\}}(\Sigma_L,X)$ dans l'algèbre $\mathcal{A}$.

Une signature d'appartenance étendue logiquement devrait être envisagée comme une signature d'appartenance normale qui contient quelques genres distingués et quelques sortes distinguées; les symboles prédicatifs sont utilisés seulement pour spécifier les genres et les sortes distingués.

Les algèbres étendues logiquement sont des algèbres d'appartenance qui correspondent aux signatures d'appartenance étendues logiquement de sorte que la signification des sortes qui représentent des prédicats est celle présentée avant.

Définition 4.2 Soit $\Omega_L = (\mathcal{B}, \leq' ,\Sigma,\Sigma_L,\pi)$ une signature d'appartenance étendue logiquement. Une Ω_L-*algèbre* est une algebre d'appartenance $\mathcal{A} = (A,\Sigma^A,\Pi_A)$ de sorte que:

- $\Pi_A(s_{p,b}) \subseteq \Pi_A(s_{p,b'})$, pour tout $p \in \Sigma_L$ et $b,b' \in B - \{0'\}$ avec $b' \leq' b$.
- $\Pi_A(s_{p,b}) \cap \Pi_A(s_{p,b'}) = \emptyset$, pout tout $p \in \Sigma_L$ et $b,b' \in B - \{0'\}$ de sorte qu'il y a $x \in B$ avec $x \prec' b$ et $x \prec' b'$.

Une algèbre attribue une signification à une signature en associant un ensemble de données à chaque genre et une opération (fonction) à chaque symbole fonctionnel. Par conséquent, une algèbre définit un type de données concret.

Par la *réduction/abstraction d'un type de données* nous comprenons la réduction de types/domaines très grands ou infinis à de petits types/domaines. Cette réduction est faite en utilisant des congruences et des politiques d'interprétation.

Définition 4.3 Soit $\Omega_L = (\mathcal{B}, \leq' ,\Sigma,\Sigma_L,\pi)$ une signature d'appartenance étendue logiquement, $\mathcal{A} = (A,\Sigma^A,\Pi_A)$ une Ω_L-algebre, ρ une congruence sur $\mathcal{A}$ et α une politique d'interprétation sur $\mathcal{B}$. Une α-*abstraction de* $\mathcal{A}$ *par* ρ est une Ω_L-algèbre $\mathcal{D} = (A/\rho,\Sigma^{A/\rho},\Pi')$ de sorte que:

- $\Pi'(s) = \{[a]_{\rho_{\pi(s)}} \mid [a]_{\rho_{\pi(s)}} \cap \Pi_A(s) \neq \emptyset\}$, pour toute sorte s qui ne représente pas un prédicat;
- $[a]_\rho \in \Pi'(s_{p,b})$, pour un $p \in \Sigma_{L,w}$, un $a \in A_{k_w}$ et un $b \in B$, s'il y a $b' \in B$ de sorte que $b \leq' b'$ et b' est la valeur de p sur $[a]_\rho$ selon α.

Les abstractions multi-valuées des algebres d'appartenance étendues logiquement peuvent aussi être calculées par des abstractions 2-valuées comme c'était le cas avec les structures logiques. Nous pouvons utiliser les S'-congruences introduites dans [39], qui correspondent aux abstractions 2-valuées,

pour obtenir quelques α-abstractions possibles d'une Ω_L-algèbre. Les S'-congruences sont définies sur les algèbres d'appartenance et permettent aux sortes d'être définies par sous-approximation dans l'algèbre quotient.

Maintenant, on prouve l'utilité des abstractions ci-dessus en offrant des résultats de préservation. Ces résultats sont traduits de la technique d'abstraction multi-valuée pour les structures logiques.

On réduit le problème de calculer la valeur de vérité d'une formule $\phi \in \mathcal{L}^{\{\wedge,\vee,\neg\}}(\Omega_L,X)$ dans une Ω_L-algèbre $\mathcal{A}$ pour une valuation γ au calcul de la valeur de vérité d'une formule $Tr(\phi)$ dans une $(\mathcal{B},\Sigma'_L)$-structure logique $\mathcal{S}_\mathcal{A}$ pour une valuation $Tr(\gamma)$. De plus, on prouve que la réduction conserve les valeurs de vérité et ça nous autorise à réitérer tous les résultats de préservation qui concernent les structures logiques, dans le contexte des algèbres d'appartenance étendues logiquement. Par exemple, la Théorème 3.1(1) peut être réitérer ainsi:

Théorème 4.1 Soit $\Omega_L = (\mathcal{B}, \leq' ,\Sigma,\Sigma_L,\pi)$ une signature d'appartenance étendu logiquement, $\mathcal{A} = (A,\Sigma^A,\Pi_A)$ une Ω_L-algebre, ρ une congruence sur $\mathcal{A}$, α une politique d'interprétation sur $\mathcal{B}$, $\mathcal{D} = (A/\rho,\Sigma^{A/\rho},\Pi')$ une α-abstraction de $\mathcal{A}$ par ρ et $b \in B$. S'il y a un $b' \in B$ de sorte que $\alpha(x) \in \{\exists^T,\exists^T_a,\forall \mid T \subseteq\uparrow b'\}$, pour tout $x \geq b$, alors:

$$[\phi]^\mathcal{D} \geq b \Rightarrow [\phi]^\mathcal{A} \geq b'$$

pour tout $\phi \in \mathcal{L}^{\{\wedge\}}(\Omega_L,X)$. De plus, si pour tout $B' \subseteq B$,

$$\vee B' \geq b \Rightarrow (\exists x \in B')(x \geq b),$$

alors

$$[\phi]^\mathcal{D} \geq b \Rightarrow [\phi]^\mathcal{A} \geq b',$$

pour tout $\phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Omega_L,X)$.

La technique d'abstraction présentée généralise et clarifie la nature de beaucoup de techniques d'abstraction trouvées dans la littérature, telles: la technique de dupliquer les symboles prédicatifs [23, 32, 6], "shape analysis" [79, 85], l'abstraction par prédicats [48, 33, 92], l'approche de McMillan [73] etc. Par exemple, il est montré que la technique de dupliquer les symboles prédicatifs, qui associe deux versions à chaque formule, une, utilisée pour validation et l'autre utilisée pour réfutation, consiste en deux abstractions basées sur la même congruence. Une abstraction est utilisée conjointement avec des formules de validation (parce que ce type d'abstraction préserve la valeur de vérité 1), et l'autre est utilisé conjointement avec des formules de

réfutation (parce que ce type d'abstraction préserve la valeur de vérité 0). Par conséquent, la nature de cette technique est soulignée clairement.

Cette technique d'abstraction peut être élargie pour les types abstraits de données. Ici, les abstractions sont appliquées aux spécifications initiales au moyen des équations. Le résultat d'une telle abstraction est une spécification d'un type abstrait de données quotient qui est en fait un nouveau type abstrait de données. Par conséquent, les techniques d'analyse qui concernent les types abstraits de données peuvent être combinées avec notre technique d'abstraction et appliquées pour raisonner au sujet des types abstraits de données.

Les spécifications des algèbres d'appartenance étendues logiquement sont simplement des spécifications des algèbres d'appartenance où les phrases pour les symboles prédicatifs sont données séparément.

Définition 4.4 Une *spécification d'appartenance étendue logiquement* est un uplet $LSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_P)$, où:

1. $(\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ est une signature d'appartenance étendue logiquement;
2. X est un ensemble de variables, disjoint de Σ;
3. E est un ensemble de phrases sur X et la signature d'appartenance $\Omega = (\Sigma, \pi)$ qui ne contient pas les opérateurs $(\_, \ldots, \_)$ et les sortes $s_{p,b}$;
4. E_P est un ensemble de phrases de la forme:
   (a) $t : s_{p,b}$ `if` $\mathcal{C}$;
   (b) $x : s_{p,b_1}$ `if` $x : s_{p,b_2}$,

   où $p \in \Sigma_{L,w}$, t est un terme de genre k_w sur Ω et X, x est une variable de genre k_w, $b, b_1, b_2 \in B - \{0'\}$, $b_1 \prec' b_2$ et $\mathcal{C}$ est un ensemble de formules atomiques sur Ω et X. E_P contient exactement une phrase du type $(4b)$ pour chaque $p \in \Sigma_L$ et b_1, b_2 avec $b_1 \prec' b_2$.
   De plus, E_P devrait satisfaire la *condition de consistance* suivante: pour tout prédicat $p \in \Sigma_L$ et $b_1, b_2 \in B - \{0'\}$ de sorte qu'il y a $x \in B$ avec $x \prec' b_1$ et $x \prec' b_2$, il n'y a aucun terme t avec $E \cup E_P \vdash t : s_{p,b_1}$ et $E \cup E_P \vdash t : s_{p,b_2}$.

Définition 4.5 Soit $LSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_P)$ une spécification d'appartenance étendue logiquement. La *sémantique initiale* de LSp, $\mathcal{M}(LSp)$, est la classe de toutes les Ω_L-algèbres isomorphes à $\mathcal{T}_{\Omega_L, E \cup E_P}$.

Une abstraction d'une Ω_L-algèbre $\mathcal{A}$ consiste dans une congruence ρ et une politique d'interprétation qui redéfinit les symboles prédicatifs dans le quotient de $\mathcal{A}$ par ρ. Naturellement, les abstractions peuvent être appliquées aux types abstraits de données $\mathcal{M}(LSp)$ au moyen d'un de leurs représentants, et $T_{\Omega, E \cup E_P}$ est un choix convenable.

Quand une abstraction est spécifiée par un ensemble d'équations, on dit qu'il est une *abstraction équationnelle*.

Définition 4.6 Soit $LSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_P)$ une spécification d'appartenance étendue logiquement. Une *abstraction de LSp* est une paire $\Delta = (A, \alpha)$, où A est un ensemble de phrases sur Ω et X, et α est une politique d'interprétation sur $\mathcal{B}$.

Définition 4.7 Soit $LSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_P)$ une spécification d'appartenance étendue logiquement et $\Delta = (A, \alpha)$ une abstraction de LSp. La *spécification d'appartenance étendue logiquement pour l'abstraction de LSp par* Δ est $LSp_\Delta = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E \cup A, E_P^\Delta)$, où:

- $E \cup A \cup E_P^\Delta \vdash t : s_{p,b}$ si et seulement si la valeur de p sur l'ensemble des termes $\{t' | E \cup A \vdash t = t'\}$ selon α est plus grande que ou égale à b (par rapport à $\leq'$).

On peut prouver facilement que la sémantique initiale de LSp_Δ contient des algèbres qui sont des α-abstractions des algèbres de la sémantique initiale de LSp.

5 Abstractions des types de données dynamiques

Plusieurs approches pour modeler les systèmes dynamiques par des algèbres universelles ont été proposées (voir [2] pour une étude sur ce sujet). Toutes les approches sont basées sur des prédicats qui sont ajoutés d'une façon ou d'une autre à la signature, mais ils opèrent en dehors de l'algèbre. Dans l'approche qu'on propose [38] nous modelons les systèmes dynamiques (qui peuvent être décrits par des états et des transitions) par des algèbres d'appartenance, un cadre logique convenable dans lequel une grande gamme de formalismes de spécification équationnelle, totaux ou partiels, peuvent être représentés naturellement [77]. Le formalisme de l'algèbre d'appartenance est tout à fait général et expressif, il supporte des sous-sortes et des surchargements et il peut modeler les erreurs et la partialité. De plus, les spécifications des algèbres d'appartenance peuvent être réalisées efficacement dans des systèmes comme Maude [27]. Nous ajoutons aussi des prédicats aux algèbres d'appartenance comme dans les approches mentionnées dans [2], mais ils opèrent à l'intérieur de l'algèbre (y compris le prédicat de transition). Par conséquent, le formalisme algèbre-logique fonctionne d'une façon unitaire.

Pour modeler les systèmes dynamiques par des algèbres d'appartenance nous considérons des algèbres étendues logiquement avec un genre spécial `state`, un ensemble de prédicats du type `state` (définissant les propriétés des états) et un prédicat de transition du type `state state`.

Définition 5.1 Une *signature dynamique* est une signature étendue logiquement $\Omega_D = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$, où:

- l'ensemble de genres de base K' contient un genre distingué `state`;
- $\Sigma_{L,w} = \emptyset$, pour tout $w \in K'^{+} - \{\texttt{state}, \texttt{state state}\}$ et $\Sigma_{L,\texttt{state state}} = \{\rightarrow\}$. On a seulement un prédicat $\rightarrow$ du type `state state` appelé *le prédicat de transition* et quelques prédicats du type `state`, appelés *propositions atomiques*.

Le genre associé au type logique `state state` est noté par `step`.

Comme une signature dynamique est un cas particulier de signature étendue logiquement, il faudrait l'envisager comme une signature d'appartenance ordinaire qui contient deux genres distingués et quelques sortes distinguées; les symboles prédicatifs utilisés spécifient seulement les sortes distinguées.

Définition 5.2 Soit $\Omega_D = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ une signature dynamique. Une *Ω_D-algèbre dynamique* $\mathcal{A}$ est une Ω_D-algèbre étendue logiquement.

Étant donné une signature dynamique Ω_D, on définit l'ensemble de *CTL^*-formules sur Ω_D* comme l'ensemble de toutes CTL^*-formules sur l'ensemble de propositions atomiques de Ω_D. Ces formules seront interprétées sur *les structures Kripke multi-valuées associées aux Ω_D-algèbres*. Plus précisément, si $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ est une Ω_D-algèbre, alors *la structure Kripke multi-valuée associée à $\mathcal{A}$*, notée par $M(\mathcal{A})$, est le triple $(Q, \rightarrow_A, L_A)$, où:

- $Q = A_{state}$;
- $\rightarrow_A$ (q,q') est la valeur du prédicat $\rightarrow$ sur (q,q') dans l'algèbre $\mathcal{A}$;
- $L_A(q,p)$ est la valeur du prédicat p sur q dans l'algèbre $\mathcal{A}$.

Étant donné une Ω_D-algèbre $\mathcal{A}$ et une CTL^*-formule ϕ sur Ω_D, on définit $\mathcal{I}_{\mathcal{A}}(\phi,\pi) = [\phi]_{\pi}^{M(\mathcal{A})}$ et $\mathcal{I}_{\mathcal{A}}(\phi,q) = [\phi]_{q}^{M(\mathcal{A})}$, pour tout chemin π et tout état q.

Les spécifications d'algèbres dynamiques sont simplement des spécifications d'algèbres d'appartenance où les phrases d'appartenance pour les propositions atomiques et pour le prédicat de transition sont données séparément.

Définition 5.3 Une *spécification dynamique* est un uplet

$$DSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_{AP}, E_{\rightarrow}),$$

où:

1. $(\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ est une signature dynamique;
2. X est un ensemble de variables, disjoint de Σ;
3. E est un ensemble de phrases sur X et la signature d'appartenance $\Omega = (\Sigma, \pi)$ qui ne contient pas les opérateurs $(\_, \ldots, \_)$ et les sortes $s_{p,b}$;
4. E_{AP} est un ensemble de phrases d'appartenance de la forme:
   (a) $t : s_{p,b}$ `if` $\mathcal{C}$;
   (b) $x : s_{p,b_1}$ `if` $x : s_{p,b_2}$,
   où p est une proposition atomique, t est un terme du genre `state` sur Ω et X, x est une variable du genre `state`, $b, b_1, b_2 \in B - \{0'\}$, $b_1 \prec' b_2$ et $\mathcal{C}$ est un ensemble de formules atomiques sur Ω et X. E_P contient exactement une phrase du type $(4b)$ pour chaque p et $b_1 \prec' b_2$.
5. $E_{\rightarrow}$ est un ensemble de phrases d'appartenance de la forme:
   (a) $(t, t') : s_{\rightarrow,b}$ `if` $\mathcal{C}$;
   (b) $x : s_{\rightarrow,b_1}$ `if` $x : s_{\rightarrow,b_2}$,
   où (t, t') est un terme du genre `step` sur Ω et X, x est une variable du genre `step`, $b, b_1, b_2 \in B - \{0'\}$, $b_1 \prec' b_2$ et $\mathcal{C}$ est un ensemble de formules atomiques sur Ω et X. De plus, $E_{\rightarrow}$ contient exactement une phrase du type $(5b)$ pour tout $b_1 \prec' b_2$.

Les phrases qui définissent les propositions atomiques et le prédicat de transition devraient satisfaire la *condition de consistance* suivante: pour tout prédicat $p \in \Sigma_L$ et $b_1, b_2 \in B - \{0'\}$ pour lesquels il y a un $x \in B$ avec $x \prec' b_1$ et $x \prec' b_2$, il n'y a aucun terme t avec $E \cup E_{AP} \cup E_{\rightarrow} \vdash t : s_{p,b_1}$ et $E \cup E_{AP} \cup E_{\rightarrow} \vdash t : s_{p,b_2}$.

Définition 5.4 Soit $DSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_{AP}, E_{\rightarrow})$ une spécification dynamique. La *sémantique initiale* de DSp est la classe $\mathcal{M}(DSp)$ de toutes les Ω_D-algèbres isomorphes à $\mathcal{T}_{\Omega_D, E'}$, où $E' = E \cup E_{AP} \cup E_{\rightarrow}$. $\mathcal{M}(DSp)$ est appelé un *type abstrait de données dynamique*.

Si nous faisons une abstraction sur une Ω_D-algèbre $\mathcal{A}$ par une congruence ρ et deux politiques d'interprétation α_R et α_L, alors la structure Kripke associée à l'abstraction est obtenue en appliquant ρ à $M(\mathcal{A})$ et en réinterprétant le prédicat de transition selon α_R et les propositions atomiques selon α_L. Nous pouvons utiliser les rapports entre les deux structures Kripke établis dans Section 3 pour obtenir les résultats de préservation pour les types de données dynamiques correspondant.

Définition 5.5 Soit $\Omega_D = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ une signature dynamique, $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ une Ω_D-algèbre, ρ une congruence sur $\mathcal{A}$ et α_R, α_L deux politiques

d'interprétation sur $\mathcal{B}$. La (α_R,α_L)-*abstraction de* $\mathcal{A}$ *par* ρ est une Ω_D-algèbre $\mathcal{B} = (A/\rho,\Sigma^{A/\rho},\Pi')$ de sorte que:

- $\Pi'(s) = \{[a]_{\rho_{\pi(s)}} \mid [a]_{\rho_{\pi(s)}} \cap \Pi_A(s) \neq \emptyset\}$, pour toute sorte s qui ne représente pas un prédicat;
- $[a]_\rho \in \Pi'(s_{\rightarrow,b})$, pour un $a \in A_{\texttt{step}}$ et un $b \in B$, s'il y a $b' \in B$ de sorte que $b \leq' b'$ et b' est la valeur de $\rightarrow$ sur $[a]_\rho$ selon α_R;
- $[a]_\rho \in \Pi'(s_{p,b})$, pour un $p \in \Sigma_{L,\texttt{state}}$, un $a \in A_{\texttt{state}}$ et un $b \in B$, s'il y a $b' \in B$ de sorte que $b \leq' b'$ et b' est la valeur de p sur $[a]_\rho$ selon α_L;

Il est clair dont que les abstractions de types de données dynamiques correspondent aux abstractions de structures Kripke multi-valuées.

Théorème 5.1 Soit $\Omega_D = (\mathcal{B}, \leq' ,\Sigma,\Sigma_L,\pi)$ une signature dynamique et $\mathcal{A} = (A,\Sigma^A,\Pi_A)$ une Ω_D-algèbre. Si $\mathcal{B}$ est une (α_R,α_L)-abstraction de $\mathcal{A}$ par ρ, alors $M(\mathcal{B})$ est une (α_R,α_L)-abstraction de $M(\mathcal{A})$ par ρ.

La théorème ci-dessus nous permet d'étendre tous les résultats de préservation qui sont valables pour les abstractions de structures Kripke multi-valuées aux abstractions de types de données dynamiques. Par exemple, la Théorème 3.4 devient:

Théorème 5.2 Soit $\Omega_D = (\mathcal{B}, \leq' ,\Sigma,\Sigma_L,\pi)$ une signature dynamique, $D \subseteq B$, $b \in B$, $\mathcal{B}$ une (α_R,α_L)-abstraction d'une Ω_D-algèbre $\mathcal{A}$ par une congruence ρ, et ϕ (ψ) une $\forall CTL^*_+$-formule d'état (de chemin) sur Ω_D. Si

1. $(\alpha_R(b) = \exists^S \;\Rightarrow\; S \cap D = \emptyset) \;\wedge\; (\alpha_R(b) = \exists^S_a \;\Rightarrow\; S \cap D = b\downarrow \cap D = b\uparrow \cap D = \emptyset)$, pour tout $b \in B - D$;
2. $(\alpha(d) = \exists^S \;\Rightarrow\; S \subseteq b\uparrow) \;\wedge\; (\alpha(d) = \exists^S_a \;\Rightarrow\; S \cup d\downarrow \cup d\uparrow \subseteq b\uparrow)$, pour tout $\alpha \in \{\alpha_R,\alpha_L\}$ et $d \geq b$;
3. pour tout sous-ensemble B' de B, $\vee B' \geq b$ implique $b' \geq b$, pour un $b' \in B'$,

alors

$$(\forall q \in A_{\texttt{state}})(\mathcal{I}_{\mathcal{B}}(\phi,[q]) \geq b \;\;\Rightarrow\;\; \mathcal{I}_{\mathcal{A}}(\phi,q) \geq b)$$

et

$$(\forall \pi_2 \in \Pi(M(\mathcal{B}),D))(\mathcal{I}_{\mathcal{B}}(\psi,\pi_2) \geq b \;\;\Rightarrow\;\; (\forall \pi_1 \in C_{M(\mathcal{A})}(\pi_2))(\mathcal{I}_{\mathcal{A}}(\psi,\pi_1) \geq b)).$$

Les abstractions peuvent être appliquées naturellement aux types abstraits de données dynamiques $\mathcal{M}(DSp)$ au moyen d'un de leurs représentant, et $T_{\Omega_D,E\cup E_{AP}\cup E_{\rightarrow}}$ est un choix convenable. Les abstractions sont spécifiées par des équations et, par conséquent, elles sont des *abstractions équationnelles*.

Définition 5.6 Soit $DSp = (\mathcal{B}, \leq' ,\Sigma,\Sigma_L,\pi,X,E,E_{AP},E_{\rightarrow})$ une spécification dynamique. Une abstraction de DSp est un triple $\Delta = (A,\alpha_R,\alpha_L)$, où A

est un ensemble de phrases sur Ω et X, et α_R et α_L sont deux politiques d'interprétation sur $\mathcal{B}$.

Définition 5.7 Soit $DSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_{AP}, E_{\rightarrow})$ une spécification dynamique et $\Delta = (A, \alpha_R, \alpha_L)$ une abstraction de DSp. La *spécification dynamique pour l'abstraction de DSp par* Δ est $DSp_\Delta = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E \cup A, E^\Delta_{AP}, E^\Delta_{\rightarrow})$, où:

- $E \cup A \cup E^\Delta_{AP} \vdash t : s_{p,b}$ ssi la valeur de p sur l'ensemble de termes $\{t' | E \cup A \vdash t = t'\}$ selon α_L est plus grande que ou égale à b (par rapport à $\leq'$).
- $E \cup A \cup E^\Delta_{\rightarrow} \vdash t : s_{p,b}$ ssi la valeur de $\rightarrow$ sur l'ensemble de termes $\{t' | E \cup A \vdash t = t'\}$ selon α_R est plus grande que ou égale à b (par rapport à $\leq'$).

La définition de DSp_Δ implique que la sémantique initiale de DSp_Δ contient des algèbres dynamiques qui sont des (α_R, α_L)-abstractions des algèbres dynamiques de la sémantique initiale de DSp.

6 Des techniques de raffinement d'abstractions

Le problème principal qui survient quand on utilise les abstractions est de trouver l'abstraction convenable ou de raffiner une abstraction déjà existante pour en obtenir une meilleure [65, 86, 56, 22, 74, 3, 68]. Dans le dernier chapitre de la thèse, nous prouvons que les techniques d'abstraction que nous avons introduites pour les types de données sous l'interprétation 3-valuée Kleene [90], peuvent être utilisées dans une procédure de raffinement.

Nous montrons qu'on peut définir une abstraction d'un type de données à partir d'une autre abstraction du même type de données. En fait, ça implique la construction d'une algèbre quotient à partir d'une algèbre quotient et une congruence. Une telle réduction dans deux pas devrait être définissable également par une réduction dans un seul pas.

Si Ω_L est une signature étendue logiquement, $\mathcal{A}$ une Ω_L-algèbre et θ, ρ deux congruences sur $\mathcal{A}$, on dit que θ est *plus fine que* ρ si $\theta \subseteq \rho$. La Figure 4 donne une vision illustrée de la propriété "plus fine que" entre congruences.

On peut prouver facilement (voir aussi [76]) que la relation binaire ρ/θ donnée par

$$(\rho/\theta)_k = \{([a]_{\theta_k}, [b]_{\theta_k}) | (a,b) \in \rho_k\},$$

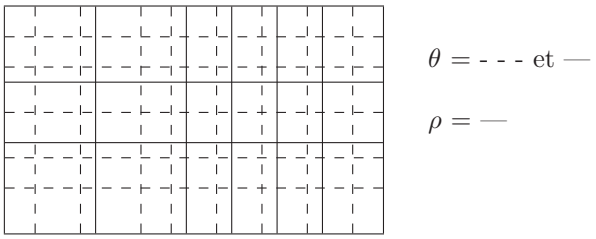

FIG. 4 – θ *est plus fine que* ρ

pour tout genre k, est une congruence sur $\mathcal{A}/\theta$ toutes les fois que θ et ρ sont des congruences sur $\mathcal{A}$ de sorte que $\theta \subseteq \rho$.

Théorème 6.1 Soit $\Omega_L = (\mathcal{B}_3, \leq', \Sigma, \Sigma_L, \pi)$ une signature étendue logiquement, $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ une Ω_L-algèbre, $\alpha \in \{\alpha_1, \alpha_2, \alpha_3\}$, $\mathcal{B}_1$ une α-abstraction de $\mathcal{A}$ par une congruence θ, et $\mathcal{B}_2$ une α-abstraction de $\mathcal{A}$ par une congruence ρ. Si θ est plus fine que ρ alors il y a $\mathcal{B}$ une α-abstraction de $\mathcal{B}_1$ par une congruence δ de sorte que $\mathcal{B} \cong \mathcal{B}_2$ et $\mathcal{B}_2 \cong \mathcal{B}$.

Une technique très populaire pour découvrir automatiquement des abstractions est le raffinement guidé par contre-exemple (en anglais, counterexample guided abstraction refinement) [22] qui commence par une abstraction initiale et après, elle utilise les contre-exemples trouvés dans le processus de la vérification pour raffiner l'abstraction actuelle. Les différentes techniques de raffinement guidé par contre-exemple, introduites dans la littérature sont basées en principal sur l'abstraction par prédicats.

On introduit dans ce chapitre une procédure de raffinement guidé par contre-exemple pour les abstractions équationnelles de structures Kripke 2-valuées. Ces structures sont représentées par des théories de réécriture comme dans [78] mais la représentation des propositions atomiques est améliorée pour ne pas conduire à des abstractions inutiles.

La procédure de raffinement ajoute des formules atomiques aux conditions des équations qui forment l'abstraction. Par opposition à l'approche qui utilise l'abstraction par prédicats où nous ajoutons au moins un prédicat, il est possible que le nombre d'états de l'abstraction actuelle ne double pas. Cela impliquera tout comme on prouve par un exemple complexe, que cette nouvelle procédure de raffinement peut construire des abstractions plus petites.

7 Conclusions

Les techniques d'abstraction, souvent reposant sur l'interprétation abstraite de Cousot [30], fournissent une méthode pour exécuter symboliquement

les systèmes en utilisant le domaine abstrait à la place du domaine concret. Dans cette thèse, on introduit des abstractions pour des logiques sous des interprétations multi-valuées. On offre des résultats de préservation pour la logique de premier ordre, la logique temporelle et la logique temporelle de la connaissance. Nous montrons comment l'abstraction peut être utilisée pour résoudre le problème de sûreté dans les systèmes de protections qui modèlent les politiques du contrôle d'accès dans une étude de cas.

L'usage d'abstraction dans le contexte des types de données est examiné également. La technique est étendue aux types abstraits de données, où les abstractions s'appliquent aux spécifications initiales à l'aide des équations. Pour raisonner au sujet des systèmes dynamiques, nous introduisons les types de données dynamiques et nous étendons la technique d'abstraction antérieure à ce cas.

Finalement, nous prouvons que les techniques d'abstraction, que nous avons introduites pour les types de données sous l'interprétation 3-valuée Kleene [90], peuvent être utilisées dans une procédure de raffinement. De plus, on introduit une procédure de raffinement guidé par contre-exemple pour les abstractions équationnelles de structures Kripke 2-valuées et on prouve par un exemple complexe, que cette nouvelle procédure de raffinement peut construire de meilleures abstractions.

Références

[1] P. E. Ammann and R. Sandhu. Extending the creation operation in the schematic protection model. In *Proceedings of the 6th Annual Computer Security Applications Conference*, pages 304–348, 1990.

[2] E. Astesiano, M. Broy, and G. Reggio. Algebraic specification of concurrent systems. In E. Astesiano, B. Krieg-Bruckner, and H.-J. Kreowski, editors, *IFIP WG 1.3 Book on Algebraic Foundations of System Specification.* Springer Verlag, 1999.

[3] T. Ball, B. Cook, S. Das, and S. K. Rajamani. Refining approximations in software predicate abstraction. In K. Jensen and A. Podelski, editors, *TACAS*, volume 2988 of *Lecture Notes in Computer Science*, pages 388–403. Springer, 2004.

[4] T. Ball, A. Podelski, and S. K. Rajamani. Boolean and cartesian abstraction for model checking C programs. In Margaria and Yi [71], pages 268–283.

[5] N. Belnap. A useful four-valued logic. In Donn and Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 30–56. Reidel, 1977.

[6] M. Bidoit and A. Boisseau. Algebraic abstractions. In M. Cerioli and G. Reggio, editors, *WADT*, volume 2267 of *Lecture Notes in Computer Science*, pages 21–47. Springer, 2001.

[7] L. Bolc and P. Borowik. *Many-valued Logics*. Springer-Verlag, 1992.

[8] G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In Halbwachs and Peled [52], pages 274–287.

[9] G. Bruns and P. Godefroid. Temporal logic query checking. In *LICS*, pages 409–417, 2001.

[10] G. Bruns and P. Godefroid. Model checking with multi-valued logics. In J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, editors, *ICALP*, volume 3142 of *Lecture Notes in Computer Science*, pages 281–293. Springer, 2004.

[11] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[12] J. A. Brzozowski and C. J. H. Seger. A unified framework for race analysis of asynchronous networks. *Journal of the ACM*, 36(1):20–45, 1989.

[13] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.

[14] D. R. Chase, M. N. Wegman, and F. K. Zadeck. Analysis of pointers and structures. In *PLDI*, pages 296–310, 1990.

[15] M. Chechik, B.Devereux, and S. M. Easterbrook. Implementing a multi-valued symbolic model checker. In Margaria and Yi [71], pages 404–419.

[16] M. Chechik, B. Devereux, S. Easterbrook, and A. Gurfinkel. Multi-valued symbolic model-checking. *ACM Transactions on Software Engineering Methodologies.*, 12(4):371–408, 2003.

[17] M. Chechik, A. Gurfinkel, and B. Devereux. χ-check: A multi-valued model-checker. In E. Brinksma and K. G. Larsen, editors, *CAV*, volume 2404 of *Lecture Notes in Computer Science*, pages 505–509. Springer, 2002.

[18] M. Chechik and W. MacCaull. CTL model-checking over logics with non-classical negations. In *ISMVL*, pages 293–. IEEE Computer Society, 2003.

[19] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In D. Kozen, editor, *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.

[20] E. M. Clarke, T. Filkorn, and S. Jha. Exploiting symmetry in temporal logic model checking. In Courcoubetis [28], pages 450–462.

[21] E. M. Clarke, O. Grumberg, H. Hiraishi, S. Jha, D. E. Long, K. L. McMillan, and L. A. Ness. Verification of the Futurebus+ cache coherence protocol. In D. Agnew, L. J. M. Claesen, and R. Camposano, editors, *CHDL*, volume A-32 of *IFIP Transactions*, pages 15–30. North-Holland, 1993.

[22] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM*, 50(5):752–794, 2003.

[23] E. M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994.

[24] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 2000.

[25] E. M. Clarke and R. P. Kurshan, editors. *Computer Aided Verification, 2nd International Workshop, CAV '90, New Brunswick, NJ, USA, June 18-21, 1990, Proceedings*, volume 531 of *Lecture Notes in Computer Science*. Springer, 1991.

[26] E. M. Clarke, D. E. Long, and K. L. McMillan. Compositional model checking. In *LICS*, pages 353–362. IEEE Computer Society, 1989.

[27] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. F. Quesada. Maude: specification and programming in rewriting logic. *Theoretical Computer Science*, 285(2):187–243, 2002.

[28] C. Courcoubetis, editor. *Computer Aided Verification, 5th International Conference, CAV '93, Elounda, Greece, June 28 - July 1, 1993, Proceedings*, volume 697 of *Lecture Notes in Computer Science*. Springer, 1993.

[29] C. Courcoubetis, M. Y. Vardi, P. Wolper, and M. Yannakakis. Memory efficient algorithms for the verification of temporal properties. In Clarke and Kurshan [25], pages 233–242.

[30] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM Symposium on Principles of Programming Languages*, pages 238–252, 1977.

[31] D. Dams. *Abstract Interpretation and Partial Refinement for Model Checking*. PhD thesis, Technische Universitat Eindhoven, 1996.

[32] D. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems. *ACM Transaction on Programming Languages and Systems*, 19(2):253–291, 1997.

[33] S. Das, D. L. Dill, and S. Park. Experience with predicate abstraction. In Halbwachs and Peled [52], pages 160–171.

[34] J. Dunn. A comparative study of various model-theoretic treatments of negation: a history of formal negation. In D. Gabbay and H. Wansing, editors, *What is Negation*. Kluwer Academic Publishers, 1999.

[35] S. M. Easterbrook and M. Chechik. A framework for multi-valued reasoning over inconsistent viewpoints. In *ICSE*, pages 411–420. IEEE Computer Society, 2001.

[36] E. A. Emerson and A. P. Sistla. Symmetry and model checking. In Courcoubetis [28], pages 463–478.

[37] C. Enea. Unifying decidability results on protection systems using simulations. In T. Dimitrakos, F. Martinelli, P. Y. A. Ryan, and S. A. Schneider, editors, *Formal Aspects in Security and Trust*, volume 3866 of *Lecture Notes in Computer Science*, pages 96–111. Springer, 2005.

[38] C. Enea and C. Dima. Abstractions of multi-agent systems. In H. D. Burkhard, G. Lindemann, R. Verbrugge, and L. Zsolt Varga, editors, *CEEMAS*, volume 4696 of *Lecture Notes in Computer Science*, pages 11–21. Springer, 2007.

[39] C. Enea and F. L. Tiplea. Abstractions of dynamic data types. *Acta Informatica*. submitted.

[40] C. Enea and F. L. Tiplea. Multi-valued abstractions. *Acta Informatica*. submitted.

[41] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[42] J. C. Fernandez, C. Jard, T. Jéron, and C. Viho. Using on-the-fly verification techniques for the generation of test suites. In R. Alur and T. A. Henzinger, editors, *CAV*, volume 1102 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 1996.

[43] M. Fitting. Many-valued modal logics I. *Fundamenta Informaticae*, 15(3-4):235–254, 1991.

[44] M. Fitting. Many-valued modal logics II. *Fundamenta Informaticae*, 17(1-2):55–73, 1992.

[45] B. R. Gaines. Logical foundations for database systems. *International Journal of Man-Machine Studies*, 11(4):481–500, 1979.

[46] M. Ginsberg. Multivalued logics. a uniform approach to inference in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.

[47] P. Godefroid and D. Pirottin. Refining dependencies improves partial-order verification methods (extended abstract). In Courcoubetis [28], pages 438–449.

[48] S. Graf and H. Saïdi. Construction of abstract state graphs with PVS. In O. Grumberg, editor, *CAV*, volume 1254 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 1997.

[49] S. Graf and B. Steffen. Compositional minimization of finite state systems. In Clarke and Kurshan [25], pages 186–196.

[50] O. Grumberg and D. E. Long. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems*, 16(3):843–871, 1994.

[51] A. Gurfinkel and M. Chechik. Multi-valued model checking via classical model checking. In R. M. Amadio and D. Lugiez, editors, *CONCUR*, volume 2761 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 2003.

[52] N. Halbwachs and D. Peled, editors. *Computer Aided Verification, 11th International Conference, CAV '99, Trento, Italy, July 6-10, 1999, Proceedings*, volume 1633 of *Lecture Notes in Computer Science*. Springer, 1999.

[53] M. A. Harrison and W. L. Ruzzo. Monotonic protection systems. In DeMillo et al., editor, *Foundations of Secure Computation*. Academic Press, 1978.

[54] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Communications of ACM*, 19(8):461–471, 1976.

[55] S. Hazelhurst. *Compositional Model Checking of Partially Ordered State Spaces*. PhD thesis, University of British Columbia, 1996.

[56] T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. Lazy abstraction. In *POPL*, pages 58–70, 2002.

[57] S. Horwitz, P. Pfeiffer, and T. W. Reps. Dependence analysis for pointer variables. In *PLDI*, pages 28–40, 1989.

[58] C. N. Ip and D. L. Dill. Better verification through symmetry. *Formal Methods in System Design*, 9(1/2):41–75, 1996.

[59] N. D. Jones and S. Muchnick. Flow analysis and optimization of LISP-like structures. In S. Muchnick and N. D. Jones, editors, *Program Flow Analysis: Theory and Applications*, pages 102–131. Prentice-Hall, 1981.

[60] N. D. Jones and S. S. Muchnick. A flexible approach to interprocedural data flow analysis and programs with recursive data structures. In *POPL*, pages 66–74, 1982.

[61] S. C. Kleene. *Introduction to Metamathematics*. Van Nostrand, New York, 1952.

[62] B. Konikowska and W. Penczek. Reducing model checking from multi-valued CTL* to CTL*. In *Proceedings of the 13th International Conference on Concurrency Theory (CONCUR2002)*, volume LNCS 2421, pages 226–239, 2002.

[63] B. Konikowska and W. Penczek. On designated values in multi-valued CTL* model checking. *Fundamenta Informaticae*, 60(1-4):211–224, 2004.

[64] B. Konikowska and W. Penczek. Model checking for multivalued logic of knowledge and time. In H. Nakashima, M. P. Wellman, G. Weiss, and P. Stone, editors, *AAMAS*, pages 169–176. ACM, 2006.

[65] R. P. Kurshan. *Computer-Aided Verification of Coordinating Processes.* Princeton University Press, 1994.

[66] J. R. Larus and P. N. Hilfinger. Detecting conflicts between structure accesses. In *PLDI*, pages 21–34, 1988.

[67] R. J. Lipton and L. Snyder. A linear time algorithm for deciding subject security. *Journal of the ACM*, 24(3):455–464, 1977.

[68] A. Loginov, T. Reps, and M. Sagiv. Abstraction refinement via inductive learning. In *Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05), LNCS 3576*, pages 519–533, Edinburgh, Scotland, UK, 2005.

[69] Z. Lotfi. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[70] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems. Specification.* Springer-Verlag, 1992.

[71] T. Margaria and W. Yi, editors. *Tools and Algorithms for the Construction and Analysis of Systems, 7th International Conference, TACAS 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genova, Italy, April 2-6, 2001, Proceedings*, volume 2031 of *Lecture Notes in Computer Science*. Springer, 2001.

[72] K. L. McMillan. *Symbolic Model Checking.* Kluwer Academic, 1993.

[73] K. L. McMillan. Verification of infinite state systems by compositional model checking. In L. Pierre and T. Kropf, editors, *CHARME*, volume 1703 of *Lecture Notes in Computer Science*, pages 219–234. Springer, 1999.

[74] K. L. McMillan and N. Amla. Automatic abstraction without counterexamples. In *Proceedings of the 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03), LNCS 2619*, pages 2–17, Warsaw, Poland, 2003.

[75] K. L. McMillan and J. Schwalbe. Formal verification of the Gigamax cache consistency protocol. In N. Suzuki, editor, *Shared Memory Multiprocessing.* The MIT Press, 1992.

[76] K. Meinke and J. V. Tucker. Universal algebra. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science vol. 1*, pages 189–411. Oxford University Press, 1993.

[77] J. Meseguer. Membership algebra as a logical framework for equational specification. In *Proceedings of WADT 97*, volume LNCS 1376, pages 18–61, 1998.

[78] J. Meseguer, M. Palomino, and N. Martí-Oliet. Equational abstractions. In *Proceedings of CADE 03*, volume LNCS 1376, pages 2–16, 2003.

[79] F. Nielson, H. R. Nielson, and C. Hankin. *Principles of Program Analysis*. Springer-Verlag, 1999.

[80] D. Peled. Combining partial order reductions with on-the-fly model-checking. In D. L. Dill, editor, *CAV*, volume 818 of *Lecture Notes in Computer Science*, pages 377–390. Springer, 1994.

[81] J. Plevyak, A. A. Chien, and V. Karamcheti. Analysis of dynamic structures for efficient parallel execution. In U. Banerjee, D. Gelernter, A. Nicolau, and D. A. Padua, editors, *LCPC*, volume 768 of *Lecture Notes in Computer Science*, pages 37–56. Springer, 1993.

[82] J. P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In M. Dezani-Ciancaglini and U. Montanari, editors, *Symposium on Programming*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 1982.

[83] H. Rasiowa. *An Algebraic Approach to Non-Classical Logics. Studies in Logic and the Foundations of Mathematics.* Amsterdam:North Holland, 1978.

[84] S. Sagiv, T. W. Reps, and R. Wilhelm. Solving shape-analysis problems in languages with destructive updating. *ACM Transactions on Programming Languages and Systems*, 20(1):1–50, 1998.

[85] S. Sagiv, T. W. Reps, and R. Wilhelm. Parametric shape analysis via 3-valued logic. *ACM Transactions on Programming Languages and Systems*, 24(3):217–298, 2002.

[86] H. Saidi and N. Shankar. Abstract and model check while you prove. In *Proceedings of the 11th International Conference on Computer-Aided Verification (CAV'99)*, pages 443–454, Trento, Italy, 1999.

[87] R. Sandhu. The typed access matrix model. In *Proceedings of the 13th IEEE Symposium on Research in Security and Privacy*, pages 122–136, 1992.

[88] R. S. Sandhu. The schematic protection model: its definition and analysis for acyclic attenuating schemes. *Journal of the ACM*, 35(2):404–432, 1988.

[89] J. Stransky. A lattice for abstract interpretation of dynamic LISP-like structures. *Information and Computation*, 101(1):70–102, 1992.

[90] F. L. Tiplea and C. Enea. Abstractions of data types. *Acta Informatica*, 42(8-9):639–671, 2006.

[91] A. Valmari. A stubborn attack on state explosion. In Clarke and Kurshan [25], pages 156–165.

[92] W. Visser, S. Park, and J. Penix. Using predicate abstraction to reduce object-oriented programs for model checking. In M. P. E. Heimdahl, editor, *FMSP*, pages 3–182. ACM, 2000.

[93] E. Y. B. Wang. *Analysis of Recursive Types in an Imperative Language*. PhD thesis, University of California, Berkeley, 1994.