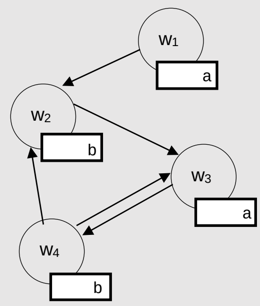
logique et REPRÉSENTATION des connaissances

Compte-rendu du cinquième TME : visite de courtoisie à LoTREC, appliqué cette fois à la logique modale.

24 octobre 2019

R////// M///////, L////// S///////



↑ Notre modèle de Kripke M, défini deux heures plus tôt :
$W = \{w_1,w_2,w_3,w_4\}$
$R = \{(w_1,w_2),(w_2,w_3),(w_3,w_4),(w_4,w_3),(w_2,w_2)\}$
$\pi(a) = \{w_1,w_3\}$
$\pi(b) = \{w_2,w_4\}$

Vérification de modèles dans la logique K

Pour chaque formule du premier exercice, on suit la procédure suivante :

edit rule: ExampleOfModelAndFormula ≫

createNewNode w1
createNewNode w2
createNewNode w3
createNewNode w4
link w1 w2 R
link w2 w3 R
link w3 w4 R
link w4 w3 R
link w4 w2 R
add w1 A
add w3 A
add w2 B
add w4 B
add w isItTrue ϕ ≫ click: Build Premodels

-- où w est un monde fixé et ϕ la formule à vérifier.

1

Mise à part la dernière ligne, la majeure partie de ce corps de règle sert à définir le modèle M représenté ci-dessus. Notre but est de vérifier onze assertions de type $M, w \models \phi$; pour ce faire et pour chacune, LoTREC crée à gauche une structure graphique qui, selon notre lecture, indique respectivement les résultats suivants :

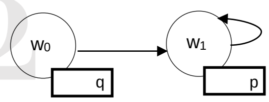
Il est vrai que $M, w_1 \models a$.
Il n'est pas vrai que $M, w_1 \models b$.
Il n'est pas vrai que $M, w_1 \models \Diamond a$.
Il est vrai que $M, w_1 \models \Box b$.
Il n'est pas vrai que $M, w_4 \models \Box b \sqcap \Box a$.
Il est vrai que $M, w_4 \models \Diamond b \sqcap \Diamond a$.
Il n'est pas vrai que $M, w_4 \models \Box b \sqcup \Box a$.
Il n'est pas vrai que $M, w_1 \models \Box\Diamond b$.
Il est vrai que $M, w_3 \models \Diamond\Diamond\Diamond a$.
Il n'est pas vrai que $M, w_3 \models \Box\Box\Box a$.
Il est vrai que $M, w_2 \models \Diamond(a \rightarrow \Box b)$.

Ces onze résultats sont justes, ils correspondent à ce qui est attendu.

Le graphe généré pour chaque exécution correspond en fait à une recherche d'information dans le modèle.
→ D'abord, le monde défini comme ancre écope d'une étiquette portant la mention *isItTrue (ϕ)?*.
→ Puis, le parser entre en action. Chaque fois qu'il rencontre un ◇ (noté *pos*), il se tourne vers les mondes accessibles depuis l'ancre et y vérifie la sous-formule placée sous la portée du ◇. Cette vérification est signalée par l'apparition d'une nouvelle étiquette au niveau des mondes concernés : par exemple, si je m'intéresse à *isItTrue (pos ψ)?* en w_1, LoTREC affecte une étiquette *isItTrue (ψ)?* à w_2, seul monde accessible depuis w_1.
→ Les résultats des vérifications remontent pour interprétation. *isItTrue (pos ψ)*? sera vrai en w_1 si *isItTrue(ψ)*? a renvoyé *Yes* dans au moins l'un des mondes accessibles. Ici, ce n'était pas le cas. On lit donc, sur l'étiquette de w_1, la mention *isItTrue (pos ψ)? [No]*. C'est fini : LoTREC nous a donné sa réponse.

À noter que le fonctionnement est le même pour □ (noté *nec*), à cela près qu'il faut que <u>tous</u> les mondes accessibles valident leur *isItTrue* (ψ)? pour valider □ψ.

2



En éditant toujours cette même règle "ExampleOfModelAndFormula", LoTREC nous autorise à définir n'importe quel ensemble de mondes et à les relier comme on le souhaite avant d'effectuer les vérifications désirées.
Par exemple, le modèle M_0 défini ci-contre contient au moins un monde w tel que $M_0, w \models \neg p \wedge \Box\Diamond p$, et un monde w tel que $M_0, w \models q \wedge \Diamond^n \neg q$, où la puissance représente des modalités ◇ successives. En particulier, w_1 vérifie la première règle, w_0 la seconde.

Il n'était pas possible de vérifier ces deux propriétés avec un modèle à un seul monde. A supposer qu'un tel modèle existe, appelons le monde ω : au moins un successeur de ω doit vérifier ¬q pour que l'application de ◇ ait une chance d'être vraie. Il ne peut s'agir de ω lui-même, puisqu'on <u>doit</u> avoir $M, \omega \models q$, ce qui suppose l'existence d'au moins deux mondes liés.

3

Mieux, LoTREC nous permet de déclarer un nouveau connecteur et d'en choisir la sémantique. Après avoir défini sa syntaxe,

add connector ≫ Name: *imp* Arity: *2* Display: _ → _ Priority: *0*

on constate que le connecteur d'implication peut être facilement implémenté avec quatre règles, à condition de respecter la séparation entre les phases ascendante et descendante de l'algorithme de vérification :

add rule: Imp_Top_Down ≫	Conditions ≫	*hasElement w imp variable A variable B*
	Actions ≫	*add w isItTrue variable A*
		add w isItTrue variable B
add rule: Imp_False_Bottom_Up ≫	Conditions ≫	*hasElement w isItTrue imp variable A variable B*
		isMarkedExpression w isItTrue variable A Yes
		isMarkedExpression w isItTrue variable B No
	Actions ≫	*markExpressions w isItTrue imp variable A variable B No*
add rule: Imp_ExFalso_Bottom_Up ≫	Conditions ≫	*hasElement w isItTrue imp variable A variable B*
		isMarkedExpression w isItTrue variable A No
	Actions ≫	*markExpressions w isItTrue imp variable A variable B Yes*
add rule Imp_True_Bottom_Up ≫	Conditions ≫	*hasElement w isItTrue imp variable A variable B*
		isMarkedExpression w isItTrue variable A Yes
		isMarkedExpression w isItTrue variable B Yes
	Actions ≫	*markExpressions w isItTrue imp variable A variable B Yes*

La première de ces règles, Imp_Top_Down, sert à indiquer à LoTREC d'initier la décomposition de la formule. Etiqueter ainsi les sous-formules A et B, c'est aller chercher leurs valeurs de vérité.
Les trois autres, de type Bottom_Up, permettent de représenter la table de vérité de l'implication. L'implication est vraie sous deux cas – si la sous-formule A est fausse, ou si les sous-formules A et B sont vraies ensemble – et fausse sinon. Alors, l'application de l'une de ces trois règles va permettre de définir l'étiquette [Yes] ou [No] associée à la formule totale isItTrue $A \rightarrow B$, qui va dépendre de ce que l'on a trouvé et remonté depuis le fond de la décomposition.

Ayant ajouté ces nouvelles règles respectivement aux stratégies prédéfinies *Top_Down* et *Bottom_Up*, on vérifie facilement la formule $P \rightarrow \Box(Q \vee P)$. Appliquée à w_1 avec Q=A et P=B, elle est vraie : A est vrai dans le monde w_1, et B est vrai dans tous les mondes accessibles (w_2). En appliquant Imp_Top_Down puis Imp_True_Bottom_Up, l'implication est validée. (Si le but était bien de faire le test avec P et Q, alors $P \rightarrow \Box(Q \vee P)$ est aussi vraie ex falso… il n'y a pas d'étiquette P.)

Satisfiabilité dans la logique K

1

$\phi_0 = \Diamond p \wedge \Box\neg p$ est une formule insatisfiable. C'est une conclusion à laquelle on parvient sans la connecter à un modèle. Pour pouvoir vérifier une formule hors de tout monde, on quitte notre premier cadre de travail pour un autre, Monomodal-K. Il suffit maintenant d'entrer la formule ϕ qui nous intéresse dans le cadre prévu à cet effet – comme nous l'avions fait à la première séance. Le graphe qui est alors construit par la fonction "Satisfiability Check" de LoTREC correspond à l'application de la méthode des tableaux sur ϕ.
Pour rappel, la méthode des tableaux renvoie une structure arborescente dont au moins une feuille doit être ouverte si la formule est satisfiable. Si les connecteurs classiques $\vee$, $\wedge$ et $\neg$ sont toujours traités de la même manière, selon les règles de décomposition de la logique propositionnelle, $\Diamond$ et $\Box$ demandent en revanche à LoTREC d'imaginer des mondes possibles. La flèche qui indique la décomposition réalisée est alors étiquetée avec la mention R : on la traduit personnellement par « création d'une potentielle relation d'accessibilité ».

Par exemple, le traitement de ϕ_0 (and pos p nec not p) donne un arbre à une seule feuille, laquelle est fermée. Il est impossible dans l'absolu, <u>pour n'importe quel monde</u>, de créer un successeur où P est vrai si, en parallèle, on souhaite qu'aucun des mondes accessibles n'ait P. L'étiquette de la feuille terminale est donc de la forme P ¬P FALSE.

2

On applique le même protocole à trois autres formules :
Le traitement de $\phi_1 = p \wedge \Diamond(q \wedge \Box\neg p)$ (and P pos and Q nec not P) donne un arbre à une seule feuille, qui est ouverte. On n'a aucun problème à créer un modèle > qui contient un monde > qui vérifie cette formule. ϕ_1 est donc satisfiable.

Le traitement de $\phi_2 = (p \wedge \neg p) \vee \Diamond\Diamond\Diamond p$ (or and P not P pos pos pos P) donne un arbre à deux feuilles, l'une ouverte, l'autre fermée. La feuille fermée correspond à la première partie de la disjonction : on ne peut pas avoir p et ¬p. $p \wedge \neg p$ étant donc toujours fausse, ϕ_2 ne sera satisfiable que si $\Diamond\Diamond\Diamond p$ l'est, c'est-à-dire si l'on peut construire un modèle > qui contient un monde > qui vérifie $\Diamond\Diamond\Diamond p$. LoTREC y parvient sans contradiction, la seconde feuille reste ouverte, donc ϕ_2 est satisfiable.

Le traitement de $\phi_3 = (p \rightarrow \Diamond(q \vee \neg p)) \vee q$ (or imp P pos or Q not P Q) donne un arbre à quatre feuilles, toutes ouvertes.
En passant sur le traitement des implications et des disjonctions, qui nous est familier, on se concentre sur les deux dernières feuilles. Elles correspondent au traitement de $\Diamond(q \vee \neg p)$, qui se traduit comme une disjonction de deux possibilités pour une potentielle relation d'accessibilité : me plaçant dans un modèle « abstrait », je tente de créer un successeur au monde dans lequel je suis, avec deux options : 1) lui affecter ¬p, 2) lui affecter q. LoTREC se rend compte qu'aucune de ces deux options ne crée de contradiction. Les deux feuilles restent ouvertes.
Donc, on peut construire au moins un modèle > qui contient un monde > qui vérifie $p \rightarrow \Diamond(q \vee \neg p) \vee q$, et ϕ_3 est satisfiable.

3

Pour prouver maintenant qu'une formule est valide, par exemple $F = \Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$, il faut et il suffit que sa négation soit toujours fausse. Lancer "Satisfiability Check" sur not imp nec imp P Q imp nec P nec Q donne un graphe à deux feuilles fermées. On en déduit que la négation de F est insatisfiable, et que F est bien valide.
On remarque avec curiosité que F est en fait l'axiome K, la distribution de Kripke. Rassuré·e en ce qui concerne la justesse de nos tests, on se souvient que tout modèle de Kripke porte en lui cette propriété, et ceci est la preuve qu'elle peut être prouvée indépendamment de tout contexte.

conclusion

Ce cinquième TME nous a permis d'approfondir notre connaissance de LoTREC en l'essayant sur un autre paradigme logique, la logique modale.

En créant des modèles, des mondes et des liens entre ces mondes, nous avons d'abord pu vérifier certaines formules *in situ*, dans un cadre défini. L'usage de □ et ◊ ne diffère syntaxiquement en rien de celui de ¬ : il s'agit juste de connecteurs d'arité 1. Le fait d'avoir à définir un nouvau connecteur, l'implication, a révélé les mécanismes internes qui permettent à LoTREC d'assurer ces vérifications. Ils se déclinent en deux phases, Top Down et Bottom Up ; LoTREC doit d'abord aller le plus loin possible dans l'évaluation des sous-formules, avant de remonter vers la formule originale pour interpréter les résultats. L'évaluation des sous-formules peut impliquer de devoir explorer des mondes accessibles (si ces sous-formules contiennent des symboles ◊ et □) : selon la relation d'accessibilité définie, LoTREC se promène alors de successeur en successeur pour y vérifier ce qu'il faut.

Il est ensuite apparu que la méthode des tableaux, comme annoncé en cours, était valable au-delà de la logique propositionnelle. Grâce à elle, il a été possible de valider ou d'invalider des formules sans les ancrer dans aucun modèle, et donc de les considérer… hors du monde et hors du temps ? Pour les dire satisfiables, on sait qu'il suffit d'imaginer un modèle dans lequel elles sont vraies. Pour celles qui l'étaient, LoTREC s'est justement montré capable de créer seul des mondes et des modèles potentiels vérifiant les exigences fixées, et de conclure.
La méthode des tableaux a en particulier permis d'établir un résultat vrai pour tout modèle de Kripke, l'axiome K – en déclarant sa négation insatisfiable.
Il est assez intéressant (émouvant ?) de remarquer comme la méthode des tableaux est capable de s'appliquer contre les axiomes de la logique modale, c'est-à-dire contre les fondements même du système.