B-005

モデル検査ツール NuSMV を用いたオントロジー検証
Ontology Verification using the Model Checker NuSMV

阿部 雄貴† 鈴木 大輝† 和泉 諭‡ 小林 秀幸† 高橋 薫†
Yuki Abe Daiki Suzuki Satoru Izumi Hideyuki Kobayashi Kaoru Takahashi

1. はじめに

オントロジーの記述は人手で行うことが大半のため，その中には構文的矛盾や，意味的な間違い，一貫性が保たれていない記述が含まれている可能性がある．そのため，記述したオントロジーの内容を検証する必要がある．主なオントロジーの検証方法として，文献[1]では，オントロジーを形式化したグラフで表すことにより，オントロジーに矛盾がないか，素 (disjoint) と補集合（complement）の関係に着目し，構文に関する検証を行っている．代表的な構文的矛盾として 6 パターンを挙げ，それぞれを検証可能としている． また文献[2]では，検証システムとして PVS を用いて，OWL（Web Ontology Language）と ORL（OWL Rule Language）を PVS 用の言語へ変換し，構文的矛盾や包摂関係の検証を行っている．文献[3]では，オントロジー内の包摂関係に着目し，矛盾の発見，修復などについて検討している．これらの研究では，包摂関係や構文的矛盾の検証を可能にしているが，それぞれを区別して扱う必要があり，同時に扱うことができない．また構文的矛盾の他に，意味的矛盾が考えられる．しかし，それらに対する検証を行うことは難しい．

本稿では，ソフトウェアや組込みシステムの検証において有効なモデル検査の手法をオントロジーの検証に応用し，文献[1]で挙げられている代表的な構文的矛盾の検証に加え，意味的矛盾の検証を行う．モデル検査ツールとしては NuSMV[4]を用いている．

2. オントロジーの状態遷移モデル

本節では，オントロジーを NuSMV で扱うための検証用のモデル（以下，状態遷移モデル）について述べる．

オントロジーと状態遷移モデルはどちらもグラフ構造を基本としているが，NuSMV で扱うモデルはラベル無し状態遷移システムであるため，単純な対応付けでは元のオントロジーが持つ関連性が崩れてしまう．また，モデル検査ツールは検証内容が満たされていない場合，そのことを示す状態遷移のパスを反例として出力し，この反例を用いることで，誤りに対する修正の手掛かりを得ることができる．オントロジーを状態遷移モデルに変換する際には，この反例の出力も考慮し，変換を行っていく．

以下，変換の方法を示す．

状態遷移モデルでは，オントロジーのトリプル〈S, P, O〉を，主語を表す〈S, P〉と目的語を表す〈P, O〉の２つに分け表現する（図１）．また，クラスとインスタンスをそれぞれ，変数として同時に記述する．クラスのみを示す場合はインスタンス変数の値は empty とし，それ以外はインスタンスが所属するクラスを示す．クラスとインスタンスを別々の変数で表すことで，クラスレベルのみの検証や，クラスとインスタンスの関連性に対し，細かな検証を行うことができる．さらに，オントロジーの制約条件，クラス演算等を表現する為に，状態遷移モデルではそれぞれ制約，クラス演算子に関する変数を加える．

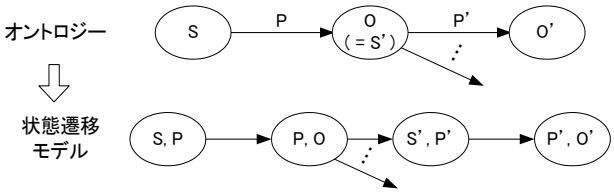

図１．オントロジーと状態遷移モデルの関係

素と補集合の関係で起こり得る矛盾において，検証後の誤りを指摘するための変数を追加する．具体的には owl:disjointWith と owl:complementOf に対し，これらの変数とは別にブール変数（シンボル）として付加し，さらにプロパティとして記述する．各々をシンボルとプロパティ変数として記述することにより，各クラスについての情報を正確に記すことができ，検証する際に，誤りの指摘を的確に行うことができる．

次に包摂関係を示すプロパティに対しての変換を行う．rdfs:subClassOf については，本来，下位クラスから上位クラスへの関係として記述されるが，状態遷移モデルで扱う際は，元のオントロジーにおいて上位クラスから下位クラスへの関係 hasSubClass へと変換する．これは，オントロジーは owl:Thing を最上位クラスとし，他の全てのクラスは owl:Thing の下位クラスとして構成されることから状態遷移モデルへ変換した際に，owl:Thing を頂点としたツリー構造となり，全状態を網羅した検証を可能とするためである．同様に rdf:type も hasType へ変換して扱う．

以上の変換を行うことにより，次節で述べるような構文的矛盾，意味的矛盾について検証を行うことができる．

3. 検証例

本節では，文献[1]で述べられている構文的矛盾として考えられるパターンの１つ（図２）を例に用いてオントロジーの検証を行う．前節に述べた変換を行い，図２のオントロジーから状態遷移モデルを生成する．

生成した状態遷移モデルを図３に示す．この状態遷移モデルを NuSMV に入力し，検証内容を CTL（Computational Tree Logic）式で記述し，モデル検査を実行する．検証内容が満たされていない場合，反例を出力する．その反例を用いて解析を行うことでオントロジーの修正を図る．

†仙台高等専門学校
Sendai National College of Technology
‡東北大学電気通信研究所/情報科学研究科
Research Institute of Electrical Communication /
Graduate School of Information Sciences, Tohoku University

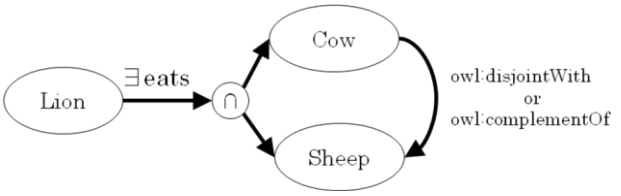

図２．構文的矛盾を含んだ例

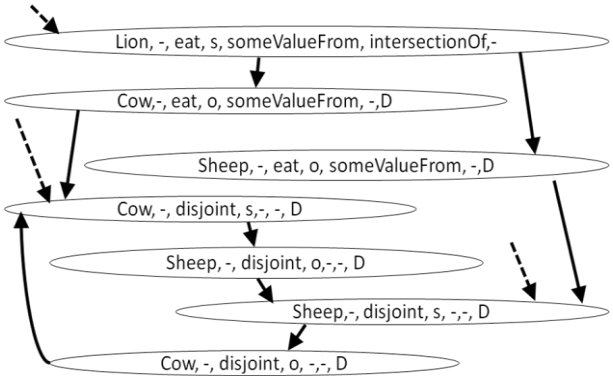

図３．図２を変換した状態遷移モデル

Loop	Step	C1	D	D2	D3	D4	D5	D6	c	i	p	r	rr	so
	0	0	0	0	0	0	0	0	Thing	empty	hasSubClass	empty	empty	s
	1	1	1	1	0	0	0	1	Carnivore	empty	hasSubClass	empty	empty	o
	2	1	1	1	0	0	0	1	Carnivore	empty	hasSubClass	empty	empty	s
	3	0	1	1	0	0	0	0	Lion	empty	hasSubClass	empty	empty	o
	4	0	1	1	0	0	0	0	Lion	empty	eat	some	intersection	s
	5	0	1	0	1	0	0	0	Sheep	empty	eat	some	intersection	o
	6	0	1	0	1	0	0	0	Sheep	empty	Disjoint	empty	empty	s
	7	0	1	0	1	0	0	0	Cow	empty	Disjoint	empty	empty	o
	8	0	1	0	1	0	0	0	Cow	empty	Disjoint	empty	empty	s
	9	0	1	0	1	0	0	0	Sheep	empty	Disjoint	empty	empty	o
	10	0	1	0	1	0	0	0	Sheep	empty	Disjoint	empty	empty	s

図４．検証結果（反例）の出力

図２に示すオントロジーは「Cow かつ Sheep であるものを Lion が食べることが少なくとも 1 つ以上存在する」ことを示している．しかし，Cow と Sheep は互いに素であり，Cow かつ Sheep なるものは存在しない．空であるものを存在制約∃（owl:someValuesFrom）してしまうことは構文的矛盾として考えられる．この矛盾に対し，検証を行う式は以下の３つであり，①から順に実行する．

[検証式]

① AG((so=s & co=intersection) -> (AX(!D)))

② AG(so=s & c=Lion & p=eat & cop=intersection -> AX(D & EX EG p=Disjoint))

③ AG!((so=s & c=Lion & p=eat & cop=intersection) & EX EX EG (D))

ここで，so=s のとき主語を，so=o のときは目的語を示す．また，クラスを示す変数 c，プロパティを示す変数 p，クラス演算を示す変数 cop を導入している．シンボル D は disjointWith で結ばれたものが存在し，同じシンボルを持っているものが互いに素であることを示している．

まず①式により，クラス演算 intersectionOf で結ばれたものの中で，互いに素である関係があるか検証を行う．ここで偽となれば矛盾が存在していることがわかる．ここで反例の解析を行い，どのクラスが矛盾しているかを見つける．次に，その矛盾しているクラスについて，どの関係で結ばれているかを解析する．②式により，「シンボル D により互いに素であるクラスがあるか」どうかを検証する．ここで真となれば矛盾が存在し，③式によりその反例（図４）を出力させ，修正することができる．

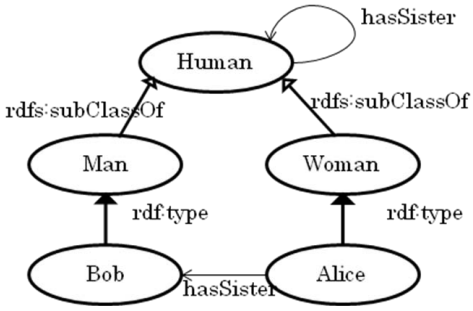

図５．意味的矛盾を含んだ例

次に意味的矛盾について述べる．図５に意味的矛盾を含んだオントロジーを示す．

Man のクラスのインスタンスが「姉妹である」と記述されているため，意味の矛盾が生じている．しかし Human の定義によれば，構文上の誤りとして見つけることが困難であり，owl:allValuesFrom の制約を記述していないなどで起こり得ることが考えられる．このような意味的矛盾は以下の検証式で検証可能である．この検証式は「男のクラスのインスタンスであり，かつ，姉妹として記述されているインスタンスとして存在するか」を検証している．ここで i!=empty はインスタンスが存在していることを示している．

[検証式]

AG!(so=s & i!=empty & EX(so=o & c=Man & p=hasSister))

4. おわりに

本稿では，モデル検査ツール NuSMV を用いて，オントロジーの構文的矛盾，意味的矛盾の検証を行うための変換，検証方法について述べた．状態遷移モデルを扱うことで制約やクラス演算を含んだオントロジーに対して検証を行うことができ，モデル検査ツールを用いたことにより，反例を利用し，矛盾の発見の手掛かりとすることができた．

今後は，変換方法に従い，オントロジーを RDF/XML 形式から NuSMV へ入力する検証ツールを実装する予定である．

参考文献

[1]S. C. Lam, D. Sleeman and W. Vasconcelos, “Graph-Based Ontology Checking”, Proceedings KCAP05 Workshop on Ontology Management: Searching, Selection, Ranking and Segmentation, pp. 33-40, 2005.

[2] J. S. Doug, Y. Feng and Y. F. Li, "Verifying OWL and ORL Ontologies in PVS", ICTAC 2004, LNCS 3407, pp. 265–279, 2005.

[3] P. Lambrix, "Repairing the Missing is-a Structure of Ontologies", Proceedings of the 4th Asian Semantic Web Conference ASWC 09, LNCS 5926, pp.76-90, 2009.

[4] A. Cimatti et, al., “NuSMV: A New Symbolic Model Verifier, ” LNCS Vol.1633, pp.495–499,1999.