アクセス制御ポリシーの生成技術及び整合性検証技術に関する研究

2009 年 7 月

大阪大学大学院工学研究科
電気電子情報工学専攻

鴨田　浩明

内容梗概

本論文は，筆者が大阪大学大学院工学研究科電気電子情報工学専攻在学中，及び株式会社エヌ・ティ・ティ・データ技術開発本部在籍中に研究開発を行ったアクセス制御ポリシーの生成技術及び整合性検証技術に関する研究成果をまとめたものであり，以下の 6 章より構成される．

第 1 章は序論であり，本論文の背景となる技術分野に関して現状を述べ，本研究の目的を明らかにする．

第 2 章では，最初に本論文で対象とするアクセス制御ポリシーの定義について述べる．その後，アクセス制御ポリシーを用いたシステム制御を実現するために必要となる 3 つの大きな技術要素である，アクセス制御ポリシー記述技術，アクセス制御ポリシー検証技術，アクセス制御ポリシー生成技術をそれぞれ述べ，技術的な課題を明らかにすることにより，本論文で議論するアクセス制御ポリシーの生成技術及び整合性検証技術の位置づけを明確にする．

第 3 章では，アクセス制御ポリシーに含まれる矛盾や冗長性を検出する技術について論じる．アクセス制御を必要とするシステムにおいて，ポリシーの設定間違いは，セキュリティ上の重大な脆弱性やシステムの性能低下に結びつくことから，アクセス制御ポリシーの整合性をシステム稼動前に検証することで安全性を証明することが求められる．そこで，アクセス制御ポリシーの整合性をタブロー法を拡張した自由変数タブロー法を用いて検証するアクセス制御ポリシー整合性検証技術について述べる．そして，シミュレーションにより，アクセス制御ポリシー整合性検証技術の実用面での有効性を検証する．

第 4 章では，アクセス制御ポリシーの内容をシステムの設定情報に反映するアクセス制御ポリシー生成技術について議論する．アクセス制御をシステム上で実現するためには，様々なシステムを整合的かつ安全に制御することが必要であり，それを手動で正確に行うことは容易ではない．そこで，オンデマンド VPN システムを事例にして，任意の二地点間で信頼性の高いセキュアな暗号化通信を実現するために必要となるシステムの構成情報を，アクセス制御ポリシーから自動的に生成する，アクセス制御ポリシー生成技術に関して述べる．そして，アクセス制御ポリシー生成技術を実装したプロタイプを用いて実証実験を行い，そこから得られた結果について考察する．

第 5 章では，設定されたアクセス制御ポリシーとシステム要件との整合性を検証する技術について議論する．第 3 章及び第 4 章で述べる技術により，整合性が検証されたア

クセス制御ポリシー通りにシステムが動作することを保証することが可能となる．しかしながら，その動作が必ずしも本来のシステム要件に合致していることは保証されていない．そこで，ドキュメント管理システムを事例に，システム要件とアクセス制御ポリシーの間に矛盾がないことを検証する，システム要件とアクセス制御ポリシーの整合性検証技術について論じる．そして，検証技術を用いることにより，実際にシステム要件を満足しているか否かを確認し得ることを検証する．

第 6 章は結論であり，本研究で得られた成果を総括する．

目　次

第 1 章

序論

インターネットは，1969 年にアメリカの 4 つの大学・研究所を結んで始まったネットワークが起源といわれている．それから 40 年，IT は著しい発展と爆発的普及を遂げ，IT システムとそれらによりネットワークを通じて提供される IT サービスは，現代社会において不可欠な存在となっている．例えば，個人においては，インターネット経由での通信販売の利用は当然のこと，金融機関のオンライン取引や電子マネーの利用など，日々の生活の中で，IT サービスを利用しない日は無いと言っても過言ではない程，IT サービスは広く普及している．また，企業などにおいては，従来より IT システムは競争力を高める戦略的ツールとして導入が進められていた．しかしながら，近年は，商取引において必要となる受発注データの電子的やりとりの標準を定めた EDI (Electronic Data Interchange) の普及など，企業間の情報連携にも IT システムを使用した方式が標準化されつつある．このことは，現在の社会構造が IT システムの利用無しに組織を運営することが困難ともいえる状況に成熟しつつあることを示している．つまり，IT システムの位置づけが，これまでの戦略的ツールから，電気・ガス・水道のようなインフラ技術へと移行しつつあることの表れでもある[7),8)]．

このように，特に企業において IT システムの役割の変化が大きい一方で，完全なインフラ技術といえるまでにコモディティ化されていない IT システムは，セキュリティ及び信頼性の観点で大きな問題を抱えている．その問題点について述べる．

図 1.1 に示すように，企業における IT システムの変化は「利用目的」「連携形態」「管理業務」という観点で整理することができる．数年前まで，企業の中で IT システムの利用目的は業務の効率化であった．つまりそれまで人手で行っていた作業のうち，自動化

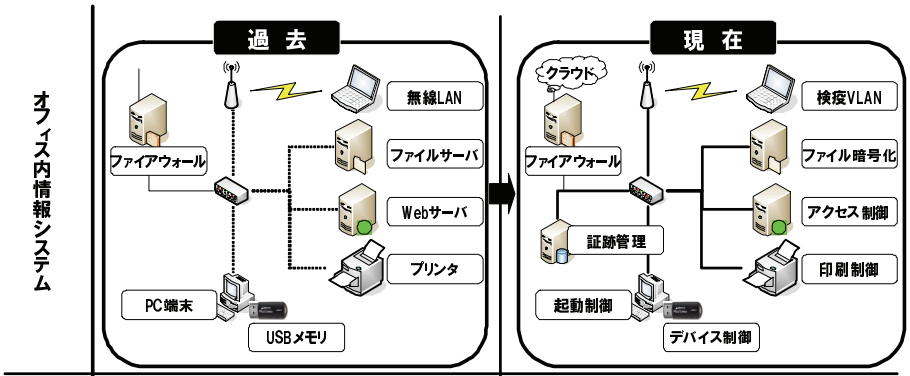

図 1.1 情報システムの変化

できるものを IT で置き換えることが主目的であった．ところが近年では，IT システムを悪用した内部犯行などの増加に伴い，コンプライアンスや内部統制が重要視されるようになった．そこで，IT システムは効率性の追求に加え，セキュリティを確保するツールとしての能力が求められるようになった．

次に連携形態に関して比較してみる．従来は紙資料を電子化し，それをシステム間で共有することで効率性を高めることが IT システム間の主な連携形態であった．つまりネットワーク層レベルでの連携ができていればある意味十分であった．しかしながら，近年はシステム間の連携技術も高度化し，単純にネットワーク上で電子ファイルを共有する形態から，電子決裁システムのように情報を加工しながらワークフローに従ってデータを社内で流通させるなど，アプリケーションレベルでのより高度な連携形態へと変化した．さらにこれら利用目的と連携形態の変化は，結果として IT システムの管理業務にも大きな変化をもたらすこととなる．これまでは，IT システム同士が疎結合の関係にあったため，いずれかの IT システムにセキュリティ上の問題やシステムダウンといった信頼性のトラブルが発生しても，その影響が他の業務へ波及する場合は少なく，業務継続性の観点におけるリスクは小さかった．しかしながら，現在は 1 箇所で発生したセキュリティインシデントやシステムダウンの影響がシステム全体に波及し，企業内に留まらず，社会活動全体に影響を与えるケースが多くなってきている[52),55),59)]．このように，IT シス

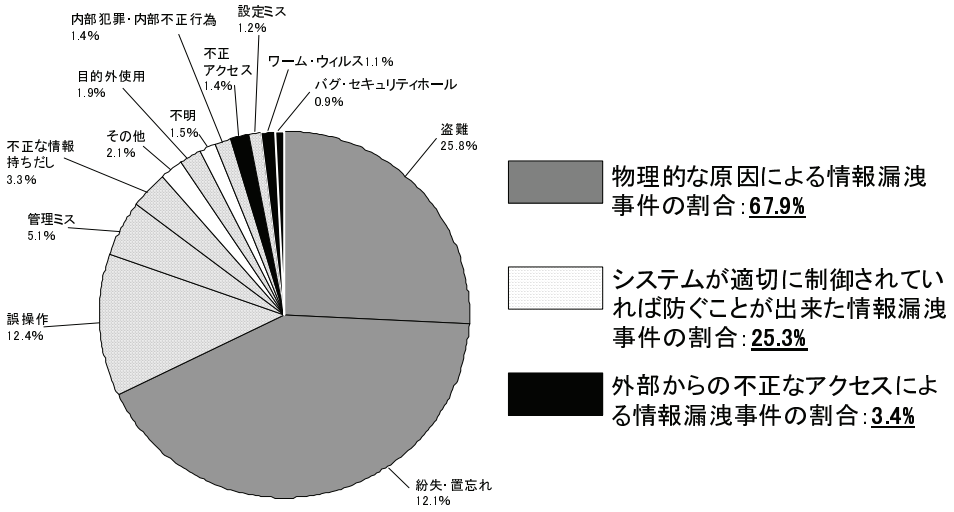

図 1.2 セキュリティインシデント (情報漏洩) の原因 (出典:JNSA「2005 年度情報セキュリティインシデントに関する調査報告 Ver.1.0」)

テムの管理業務には個々のシステム，つまり “点”の視点での管理から，連携システム全体の “線”の視点での管理が求められるように変化した.

次に，近年の IT システムの複雑化によりもたらされた問題の詳細と原因について考察する．一般的にセキュリティインシデントが発生した場合，その原因の殆どはシステムの開発過程で作り込まれたセキュリティホールなどを外部から悪用されたことに原因があると考えられがちである．同様に，システムダウンなどのシステム障害の原因も，その殆どが開発過程で作り込まれたバグが原因であると思われやすい．しかしながら，この仮説は正しくない．実際，図 1.2 及び図 1.3 に示すように，セキュリティインシデントとシステム障害の原因の大多数は別の箇所にある．図 1.2 に示すように，セキュリティインシデントの原因に関しては，セキュリティホールや外部からの攻撃などによるものは，全体の高々 3.4% に過ぎず，紛失などシステムとは無関係な物理的原因が，全体の 68%，設定を適切に行っていれば防ぐことができたインシデントが全体の約 25% であることがわかる．つまり，セキュリティインシデントに関しては，物理的な対策をしっかりと施した上で，システムの設定を間違うことなく正確に実施することにより，その殆どを防止することができることがわかる．同様に，図 1.3 は，交通機関や金融機関などの重要なインフラシステムにおいて発生したシステム停止などの障害の発生原因がどのフェーズ

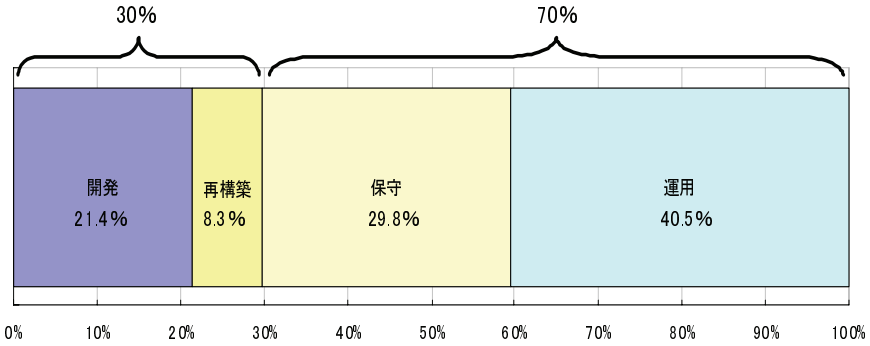

図 1.3 システム障害の原因 (出典:IPA,JUAS「重要インフラ情報システム研究会」)

にあったかを示すデータである．これによると，システム障害の原因の 70% が保守，運用の過程で発生しており，システムの開発過程で埋め込まれたバグが原因による障害は全体の高々 30% 程度に過ぎないことがわかる．つまり，システム障害を減らすには，セキュリティインシデントを減らすのと同様にシステムの保守・運用過程の作業を正確に実施することが最善の解決策であることがわかる．

このように，セキュリティやシステム障害などのトラブルを減少させるためには，既にある IT システムの機能を理解し，トラブルが発生しないように適切に管理することが重要である．しかしながら，IT システムの機能は高度化し，従来と比較して，それが適切に稼動するように初期設定や運用を行うことは非常に困難になっている．そのため，これまでのように運用管理者の経験に頼った属人的な方法で連携するシステムの設定を完全に正しく行い，システムトラブルを防止することには限界がきている．即ち，何らかの方法でシステムの設定を正しく自動的に行う技術，あるいはシステムに設定した情報の妥当性を検証する技術が，IT システムのセキュリティと信頼性を向上させるために必要とされている．

ここまで，セキュリティと信頼性の観点で，現在の IT システムが抱える問題と課題について述べてきたが，その中でも，特にセキュリティ分野においては，個人情報などのセンシティブな情報の漏洩事件が後を絶たず，金銭的な被害に加え，個人が精神的な被害を被る場合もあるなど，対策技術の早急な確立が必要とされている．図 1.2 に示したように，物理的原因以外の情報漏洩の殆どがシステム設定を適切に実施しなかったことによるものであり，具体的には，誤操作，管理ミス，不正な情報持ち出し，目的外使用，内部犯罪・内部不正行為，設定ミスなどが原因である．これらの問題の本質は，本来アクセ

スさせるべきでない資源にアクセスできるような状態を許すアクセス制御の不備にある．物理的なアクセス制御を実現する方法には，鍵の設置やガードマンの配置などいろいろな方法が考えられるが，システム的なアクセス制御の殆どは，アクセスを許可する，あるいはアクセスを禁止する条件が定義されたアクセス制御ポリシーに基づいて実現されるのが一般的である．

そこで，本論文では，個人情報も含め機密情報が漏洩すること防ぐために，システムに設定されるアクセス制御ポリシーの信頼性を確保する 3 つの技術について論じる．第一に，数が多く複雑なアクセス制御ポリシーの矛盾をシステム稼動前に静的に検証することにより，機密情報の漏洩リスクを低減させる技術について述べる．第二にネットワークセキュリティ機器のアクセス制御ポリシーを自動的に生成することにより，管理者のミスなどによる機密情報の漏洩を防止する技術について述べる．第三に，設定されたアクセス制御ポリシーにより，システムが本当に管理者の望まない動作に陥ることがないことを検証する技術について述べる．本論文の 2 章以降の構成は次の通りである．

第 2 章では，最初に本論文で対象とするアクセス制御ポリシーの定義について述べる．その後，アクセス制御ポリシーを用いたシステム制御を実現するために必要となる 3 つの大きな技術要素である，アクセス制御ポリシー記述技術，アクセス制御ポリシー検証技術，アクセス制御ポリシー生成技術を述べ，技術的な課題を明らかにすることにより，本論文で提案するアクセス制御ポリシーの生成及び整合性検証技術の位置づけを明確にする．

第 3 章では，アクセス制御ポリシーに含まれる矛盾や冗長性を検出する技術[25)–27)] について論じる．アクセス制御を必要とするシステムにおいて，ポリシーの設定間違いは，セキュリティ上の重大な脆弱性やシステムの性能低下に結びつくことから，アクセス制御ポリシーの整合性をシステム稼動前に検証することで安全性を証明することが求められる．そこで，アクセス制御ポリシーの整合性をタブロー法を拡張した自由変数タブロー法を用いて検証するアクセス制御ポリシー整合性検証技術について述べる．そして，シミュレーションにより，アクセス制御ポリシー整合性検証技術の実用面での有効性を検証する．

第 4 章では，アクセス制御ポリシーの内容をシステムの設定情報に反映するアクセス制御ポリシー生成技術[51),62),63)] について議論する．アクセス制御をシステム上で実現するためには，様々なシステムを整合的かつ安全に制御することが必要であり，それを手動で正確に行うことは容易ではない．そこで，オンデマンド VPN システムを事例にして，任意の二地点間で信頼性の高いセキュアな暗号化通信を実現するために必要となるシス

テムの構成情報を，アクセス制御ポリシーから自動的に生成する，アクセス制御ポリシー生成技術に関して述べる．そして，アクセス制御ポリシー生成技術を実装したプロタイプを用いて実証実験を行い，そこから得られた結果を考察する．

第 5 章では，設定されたアクセス制御ポリシーとシステム要件との整合性を検証する技術[13),20),24),50)] について議論する．第 3 章及び第 4 章で述べる技術により，整合性が検証されたアクセス制御ポリシー通りに，システムが動作することを保証することが可能となる．しかしながら，その動作が必ずしも本来のシステム要件に合致していることは保証されていない．そこで，ドキュメント管理システムを事例に，システム要件とアクセス制御ポリシーの間に矛盾がないことを検証する，システム要件とアクセス制御ポリシーの整合性検証技術について論じる．そして，検証技術を用いることにより，実際にシステム要件を満足しているか否かを確認し得ることを検証する．

第 6 章は結論であり，本研究で得られた成果を総括する．

第 2 章

アクセス制御ポリシーの定義と技術課題

2.1 緒言

本章では，本論文で論じる対象となるアクセス制御ポリシーの定義と，その技術的課題について述べる．

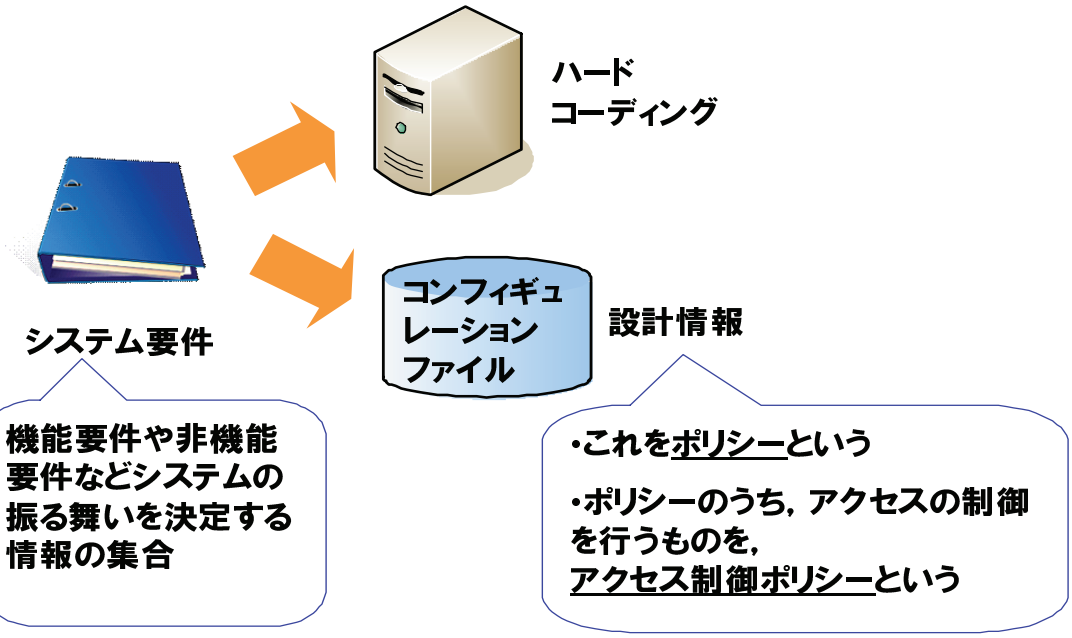

図 2.1 アクセス制御ポリシーの定義

2.2　アクセス制御ポリシーの定義

一般に IT システムは，機能要件や非機能要件などのシステム要件に基づいて設計が行われ，プログラミング工程や，試験工程を経て完成し，利用されるようになる．

このとき，要件の一部はプログラミング工程でプログラムの中に埋め込まれた形式で実現される．この実現手法を一般にハードコーディング (図 2.1 参照) と呼び，システム完成後にハードコーディングされたシステム要件を変更することは難しい．特に，システムの最終形態が，IC カードや組み込みソフトウェアのように，ハードウェアと一体となって提供される場合には，システム要件を完成後に変更することはほぼ不可能であり，製品の作り直しが求められる可能性が高い．

一方で，ハードコーディングのようにプログラムの中にシステム要件を埋め込む手法とは異なり，設定情報という形式でソフトウェアの完成後に自由にシステム要件の一部を設定・変更できるようにする実現手法もある (図 2.1 参照)．これは一般にコンフィギュレーションファイルなどと呼ばれるファイルに IT システムの設定情報を記述し，それを IT システムが動的に読み込みながら，処理をするという手法である．この手法を用いる

と，IT システムの完成後であっても，その処理内容を柔軟に変更することができるという利点があり，現在利用されているほぼすべての IT システムが，設定情報を何らかの形式で記述できるようにされている．

IT システムの設定情報は，その IT システムがどのように振る舞うかを記述したシステム要件の集合である．設定情報の種類や記述方式は数多く存在するが，本論文では，この設定情報のことを**ポリシー**と呼ぶ．即ち，ポリシーとは，IT システムの振る舞いを動的に制御するためのシステム要件の集合である．

ポリシーとして定義される情報には様々な内容がある．例えば，ユーザーインターフェースの色情報やフォント情報などもポリシーである．また，周期的に自動的にバックアップやウィルスチェックを始めたりするジョブ・スケジュール情報もポリシーである．このようにポリシーの種類は多岐にわたるが，特に重要なものの一つが，アクセス制御に関するポリシーである．例えば，ファイアウォールのパケットフィルタリングのポリシー，共有ファイルサーバー上にあるファイルの読み取りや書き込みを制御するポリシー，電子稟議などのワークフローを制御するポリシーなどである．これらアクセス制御に関するポリシーは，万が一その設定を誤ると，不正経理や機密情報の漏洩などの問題に発展する危険性があるため，セキュリティ確保の観点からポリシーの中でも特に重要性の高いものである．本論文では，このように不正や機密情報の漏洩を防ぐなどの目的で設定されたポリシーを特に，**アクセス制御ポリシー**と呼ぶ．ただし混同するおそれがない場合には，アクセス制御ポリシーのことを単にポリシーと呼ぶ場合もある．

本論文では，アクセス制御ポリシーを対象に，その技術課題を解決するための手法について論じる．そこで，次節以降では，アクセス制御ポリシーに関する技術的課題と，本論文の位置づけについて述べる．

2.3　アクセス制御ポリシーに関する技術課題

システム開発にライフサイクルがあるように，アクセス制御ポリシーにもライフサイクルを考えることができる．前節で述べたように，アクセス制御ポリシーはセキュリティの確保が目的ある．そのため，アクセス制御ポリシーのライフサイクルも，セキュリティを可視化するためのフレームワークの一つである， PDCA (Plan，Do，Check，Action) の観点で整理すると理解しやすい．そこで，本節では，PDCA の観点に照らし合わせて，アクセス制御ポリシーの技術課題について整理し，第 3 章以降で論じる解決方式との関係について述べる．

2.3.1　アクセス制御ポリシー記述技術

最初に Plan，つまりシステム稼動前の計画段階において，アクセス制御ポリシーに求められる技術について述べる．

システムをセキュリティ要件に従って制御するためには，そもそも，何らかの記述フォーマットに従って，アクセス制御ポリシーを記述する必要がある．アクセス制御ポリシーの記述方法には，グラフィカルなユーザーインターフェースを通じて記述する方法や，あるいは XML のような特定の文法に従って，テキストファイルなどに記述する方法などが存在する．いずれの方法においてもアクセス制御ポリシーを設定するシステム管理者などに，わかりやすく設定することができるインターフェースが必要となる．それを可能にする技術を総称して，本論文では**アクセス制御ポリシー記述技術**と呼ぶ．アクセス制御ポリシー記述技術に関しては，XACML[38] など，既に標準的な技術が提案されていることから，本論文における議論の対象外とする．しかしながら本論文において論じる他の技術との関係が深いため，第 3 章以降においても必要に応じて，アクセス制御ポリシー記述技術に関して補足的に説明を行う．

2.3.2　アクセス制御ポリシー生成技術

次に，Do，つまりシステムをアクセス制御ポリシーに従って動作させるために，Plan の段階で記述されたポリシーをシステムに反映する技術について述べる．

一般にアクセス制御ポリシーは，システム管理者などの人間が，当該組織などの実態を

踏まえて策定するものである．そのためシステム管理者は，前述したアクセス制御ポリシー記述技術などを利用して，人間が理解しやすい形式で，アクセス制御ポリシーを記述することとなる．しかしながら，人間にわかりやすいヒューマンフレンドリーな形式のアクセス制御ポリシーは，システムが処理可能なマシンリーダブルな形式であるとは必ずしも限らない．そこで，場合によっては，アクセス制御ポリシーをシステムが解釈可能な形式に変換する技術が必要となる．その技術を，本論文では，**アクセス制御ポリシー生成技術**と呼ぶ．

アクセス制御ポリシー生成技術が必要となる事例としては，VPN (Virtual Private Network) の構築事例をあげることができる．VPN を構築するためには，通常ルーターなどのネットワーク機器に対して，暗号化通信の方式や，暗号鍵など専門家以外には理解することが難しい複雑な設定を実施する必要がある．そのため，ある地点とある地点の VPN 接続を許可するか否かという比較的簡単なアクセス制御であっても，それを正確に実現することは難しく，アクセス制御ポリシーとして記述した内容を，アクセス制御ポリシー生成技術を用いて，専用機器が解釈できる形式に変換して反映することが求められる．本論文では，この問題を解決するアクセス制御ポリシー生成技術[51),62),63)]に関して，第 4 章で論じる．

2.3.3　アクセス制御ポリシー検証技術

Check，つまり，設定あるいは反映したアクセス制御ポリシーが本当に正しいかどうかを検証する技術について述べる．

アクセスを許可するユーザやリソースの情報など，アクセス制御ポリシーの内容は，管理対象の増加に伴い，複雑化する．そのため，アクセス制御ポリシーの中に矛盾する情報を記載したり，あるいは，結果として管理者の意図しない振る舞いをするようなアクセス制御ポリシーを設定したりする問題が発生する．そこで，アクセス制御ポリシーが正しく設定され，管理者の意図通りにシステムが動作することを検証する技術が必要となる．本論文ではこの技術を，**アクセス制御ポリシー検証技術**と呼ぶ．

アクセス制御ポリシー検証技術は，検証する対象に応じて，2 種類に分けることができる．一つは設定されたアクセス制御ポリシーそのものの中に，矛盾するポリシーや冗長なポリシーなどが含まれていないかを検証する技術である．ポリシーの中に矛盾するポリシーが含まれる場合，望まれないアクセスが許可されたり，逆に，許可されるべきアク

セスが遮断されたりするなどの問題が発生する可能性がある．また，使用されることのない冗長なアクセス制御ポリシーが含まれると，システムの性能に悪影響を及ぼす可能性がある．そこで，設定されたアクセス制御ポリシーの中に，矛盾するポリシーや冗長なポリシーが含まれている場合に，それらを検出する技術が求められる[25)–27)]．この技術を特に**アクセス制御ポリシーの整合性検証技術**と呼び，第 3 章で詳しく述べる．

もう一つが，設定されたアクセス制御ポリシーにより，システムが要件通りに正しく振る舞うかどうかを検証する技術である．設定したアクセス制御ポリシーに矛盾や冗長なポリシーが含まれないことを確認しただけでは，システムが要件通りに正しく振る舞うことを完全に保証することはできない．そこで，システムが，設定されたアクセス制御ポリシー通りに動作する場合に，どのような状態においても，システムとして本来満足するべき要件を達成するかどうかを証明する技術が求められる[13),20),24),49),50)]．この技術を特に，**システム要件とアクセス制御ポリシーの整合性検証技術**とよび，第 5 章で詳しく述べる．

最後に Action であるが，これは，Check 段階で検証したポリシーを見直し，再び Plan から Do というフェーズを実行することであるため，これまでに述べた技術の組み合わせで実現することが可能である．そのため，本論文では，Action に関する技術的課題については議論の対象外とする．

2.4 結言

本章では，本論文で論じるアクセス制御ポリシーの定義と，その技術課題に関して述べた．第 3 章〜第 5 章において，本章で述べたアクセス制御ポリシーに関する技術的課題の詳細と，その解決策について論じる．

第 3 章

アクセス制御ポリシーの整合性検証技術

3.1 緒言

第 1 章及び第 2 章において考察したように，IT システムにおいてアクセス制御ポリシーは，セキュリティを確保するための重要な役割を担っている．アクセス制御ポリシーの設定が適切に行われないと，あるユーザが本来アクセスできないはずの機密情報にアクセスできるなど，情報漏洩の問題が発生する危険性がある．あるいは逆に，本来アクセスできるべき情報にアクセスできずに，業務が停止するという可用性の問題が発生する可能性もある．さらに，設定されたアクセス制御ポリシーの中に使用されることのない冗長なポリシーが含まれていると，システムの処理が遅くなるなど，性能面の問題が発生する可能性もある．これらの問題がシステム稼動中に発生することを防止するためには，システムに設定されたアクセス制御ポリシーを分析し，お互いに矛盾するポリシーや冗長なポリシーが設定されていないことを検証する必要がある．

企業や病院，金融機関，政府機関などの多くの組織において必要とされるアクセス制御は，アクセスを許可する主体，例えば，IP アドレスやログイン ID 単位に，アクセスされる客体，例えば情報資源やネットワーク資源に対して，アクセスを許可するか禁止するかを制御することである[42)]．しかしながら，組織と組織で利用される IT システムが大きくなるに従い，管理するべき主体と客体の数も増大し，その結果，設定するべきアクセス制御ポリシーの量も増大する．さらに，IT システムごとに，異なるアクセス制御ポリシー

を設定する必要があり，同一組織内に，複数の異なるアクセス制御ポリシーが分散して存在することになる．このような状況で，アクセス制御ポリシーが IT システム全体として正しく設定されていることを検証することは容易ではない．

そこで，本章では，アクセス制御ポリシーに含まれる矛盾や冗長性を検証する技術課題について整理しその課題を解決する手法に関して論じる．次に，考案した手法を用いて，アクセス制御ポリシーの検証が可能であることを論理的に証明する．そして考案手法を用いて，アクセス制御ポリシーを実際に検証した評価結果について述べる．

3.2　アクセス制御ポリシーの整合性検証に関する技術課題

3.2.1　アクセス制御ポリシーモデルと記述言語

組織で管理される IT システムが複数有り，それぞれ異なるアクセス制御ポリシーを設定する必要があることが，アクセス制御ポリシーの整合性検証を難しくしている要因の一つである．アクセス制御ポリシーを標準化することにより，この問題を解決しようとする試みがある．ロールベースアクセス制御 (RBAC : Role Based Access Control)[14] は，米国においてその仕様が標準化されたアクセス制御モデルの一つであり，標準化によりアクセス制御ポリシーの違いによる設定の煩雑性を解消できるという利点がある．さらに RBAC ではアクセス制御の主体となる情報をロールと呼ばれる概念で集約化することにより，設定するべきアクセス制御ポリシーの数を減らす工夫が行われている．これは，人間が識別することの難しい英数字の羅列から構成されるユーザ ID 単位ではなく，例えば，“課長”や “部長”，あるいは，“営業部”や “開発部”といった，役職や所属などの単位（これを**ロール**と呼ぶ）でアクスセス制御ポリシーを設定可能にすることを意味する．

しかしながら RBAC の仕様では，アクセス制御のモデルを規定しているのみで，具体的にどのようにアクセス制御ポリシーをシステムに設定するかまでは規定していない．RBAC モデルのアクセス制御ポリシーの記述方法を規定しているのは，XACML[38] と RBAC[38] プロファイル，あるいは，X-RBAC[5] である．これらはその名称から推測できるように，XML (eXtensible Markup Language) を用いて RBAC モデルのアクセス制御ポリシーを記述する方法を規定したアクセス制御ポリシー記述言語である．他には，Ponder[12] のように RBAC モデルを拡張したアクセス制御モデルを定義すると同時に，独自のポリシー記述言語を規定したアクセス制御モデルも提案されている．本論文では特に断らない限り，Ponder といった場合は，XACML 同様にアクセス制御ポリシー記述言語の意味で用いる．また，XACML を拡張し，より汎用的なアクセス制御ポリシーの記述を可能とした提案としては，ORBAC[22] や TRBAC[4] と呼ばれるものがある．これらの共通的なアクセス制御モデルやアクセス制御ポリシー記述言語を利用することにより，異なるシステム間で共通のアクセス制御ポリシーを利用することが可能となる．

このように，標準的なアクセス制御モデルやアクセス制御ポリシー記述言語を導入することにより，アクセス制御ポリシーの管理作業の負荷をある程度低減することが可能となる．しかしながら，アクセス制御モデルの全体構造とアクセス制御ポリシー記述言

語はいずれも複雑化しているため，システム管理者にとって，設定したアクセス制御ポリシーの中に，矛盾するポリシーや，冗長なポリシーが含まれていないことを検証することは困難であり，これらを検出することのできる，アクセス制御ポリシーの整合性検証技術が必要となる．

3.2.2　アクセス制御ポリシーと矛盾検出

アクセス制御ポリシーの整合性検証技術について論じる前に，アクセス制御ポリシーの具体例について簡単に述べる．一般的にアクセス制御ポリシーは，“誰” が “何” に対してどのような “動作” を行うことを “許可される”，あるいは，“禁止される”という形式で定義される．本論文では，アクセス制御ポリシーの “誰” に該当する要素を**主体**，“何”に該当する要素を**客体**とそれぞれ呼ぶ．アクセス制御ポリシーの例を以下に示す．

アクセス制御ポリシーの例

(1) “医師”(主体) は，“カルテ”(客体) を，“書込”(動作) できる
(2) “看護師”(主体) は，“カルテ”（客体）を，“閲覧”(動作) できない

この例のように，主体，客体，動作の三つ組みで定義することのできるアクセス制御ポリシーを**認可ポリシー**とよび，認可ポリシーのうち，上の例 (1) のように許可するものを**認可 (許可) ポリシー**，(2) のように禁止するものを**認可 (禁止) ポリシー**と呼ぶ．

RBAC の仕様では，ポリシー継承という概念が規定されている．これは，一つの主体に対して設定された認可ポリシーが継承され，他の主体にも影響を及ぼすという概念である．例えば，企業において，課長と部長というロールが存在するとする．このとき，一般的に職位の低い課長に対して許可されている業務は，課長よりも職位の高い部長に対しても許可される，逆に部長に対して禁止されている業務は，課長にも禁止されることが一般的である．このように主体の階層構造に従って，アクセス制御が継承される概念が，ポリシー継承であり，現実社会において一般的に利用される概念である．

アクセス制御ポリシーの継承という概念は，一つのアクセス制御ポリシーを設定すると，暗黙的に多数のアクセス制御ポリシーを設定したのと同一の効果を生むため，アクセス制御ポリシーの記述作業を簡略化するという利点がある．一方で，継承関係により暗黙的に規定されるアクセス制御ポリシーの存在は，その解析を困難にし，お互いに矛盾

するアクセス制御ポリシーや，冗長なアクセス制御ポリシーを設定するリスクを高める．これらの矛盾するポリシーや冗長なポリシーを検出する技術がアクセス制御ポリシーの整合性検証技術である．

アクセス制御ポリシーの整合性検証技術に関しては，これまでにもいくつか提案されている．例えば，Graham[17] らは，ディシジョンテーブルを用いて継承の発生を前提としないアクセス制御ポリシーの矛盾を検出する手法を提案している．また，Strembeck[44] はポリシー継承により暗黙的に規定されたポリシーの矛盾を検出する手法を提案している．しかしながらこれらの手法はいずれも，特定のアクセス制御ポリシーモデルに限定した検出手法であり，既に数多く提案されている他のアクセス制御ポリシーモデル[9),21),37),40)] には適用することができないという制限がある．また，アクセス制御ポリシーの矛盾の有無を検出するのみで，冗長なポリシーを検出したり，その原因を特定することはできない．アクセス制御ポリシーの管理を効率化するためには，アクセス制御ポリシーの矛盾の有無を検出するのみではなく，システム管理者が検出された矛盾を直ちに解消できるように，矛盾の原因となる情報を抽出できる必要がある．

そこで，本章では，例えば XACML や Ponder のような何れかのアクセス制御ポリシー記述言語によって記述されたポリシーを，第一階述語論理式に変換することにより，論理式の推論法の一つとして広く知られている自由変数タブロー法を用いて，矛盾するポリシーと冗長なポリシーを静的に検出すると同時に，その原因を推定して導出することを可能にする手法について論じる．

3.3　アクセス制御ポリシーモデルと形式化

3.3.1　アクセス制御ポリシーの対象事例

アクセス制御ポリシーを定義しその妥当性を検証することは，様々なアプリケーションで必要とされるが，具体的なイメージをつかむために，ここでは，医療分野において利用されるオンデマンド VPN (Virtual Private Network)[57),58)] を事例として取り上げる．

オンデマンド VPN システムでは，医師や看護師などの病院の職員が他の病院のネットワークや医療機器に対して VPN 接続を行うために開発されたシステムである．大病院にしか存在しない高価な医療機器を小さな病院からアクセスして利用することができるようにしたり，あるいは，高度な技術を有する専門医による遠隔診断を実現したりすることを目的としている．病院間の VPN の接続の可否は，アクセス制御ポリシーに基づいて判断されるが，そのポリシーには，ネットワークトポロジーの変化や，病院間あるいは病院内の役職などの継承関係を扱うことができるような柔軟性が求められる．さらに遠隔診断などを行うことから，極めて高い機密性を求められる医療データを扱うため，アクセス制御システムには高度な正確性と安全性も要求される．

従来の VPN に求められるアクセス制御は一般的に，IP アドレスのようなネットワーク層の情報に基づく比較的単純な技術で実現されていた．しかしながら，オンデマンド VPN システムには，より高度なアクセス制御が必要となるため，具備されるアクセス制御モデルには様々な要件が求められる．

そこで本節では，オンデマンド VPN システムにおいて求められるアクセス制御モデルについて，具体的なアクセス制御ポリシーを例示しながら，数学的形式化を行う．

(1)	$\zeta(\mathcal{H})$	:=	$\{H_{\mathcal{H}}(i,j) : i \geq_{\mathcal{H}} j\}$
(2)	$\zeta(\texttt{Auth+}(S_1,\ T_1,\ A_1))$	:=	$\forall x(E_x \rightarrow P(S_1,T_1,A_1))$
(3)	$\zeta(\texttt{Auth-}(S_1,\ T_1,\ A_1))$	:=	$\forall x(E_x \rightarrow \neg P(S_1,T_1,A_1))$
(4)	$\zeta(\texttt{Obli+}(E_1,S_1,\ T_1,\ A_1))$	:=	$E_1 \rightarrow O(S_1,T_1,A_1)$
(5)	$\zeta(\texttt{Obli-}(E_1,S_1,\ T_1,\ A_1))$	:=	$E_1 \rightarrow R(S_1,T_1,A_1)$
(6)	Ax1	:	$\forall s,t,a(O(s,t,a) \rightarrow P(s,t,a))$
(7)	Ax2	:	$\forall s,t,a(\neg(O(s,t,a) \wedge R(s,t,a)))$
(8)	$\zeta(\text{prop1}) = \zeta(\text{prop2})$	:=	$\forall x,y,z,a(P(x,y,a) \wedge H_{\mathcal{H}}(z,x) \rightarrow P(z,y,a))$
(9)	$\zeta(\text{prop3}) = \zeta(\text{prop4})$	:=	$\forall x,y,z,a(P(x,y,a) \wedge H_{\mathcal{H}}(x,z) \rightarrow P(z,y,a))$
(10)	$\zeta(\text{prop5}) = \zeta(\text{prop6})$	:=	$\forall x,y,z,a(P(x,y,a) \wedge H_{\mathcal{H}}(y,z) \rightarrow P(x,z,a))$
(11)	$\zeta(\text{prop7}) = \zeta(\text{prop8})$	:=	$\forall x,y,z,a(P(x,y,a) \wedge H_{\mathcal{H}}(z,y) \rightarrow P(x,z,a))$
(12)	$\zeta(A_1 = \Gamma(A_2,\cdots,A_n))$	:=	$\forall x,y(P(x,y,A_1) \leftrightarrow \Gamma(P(x,y,A_2),\cdots,P(x,y,A_n)))$
(13)	$\zeta(\text{cw})$	:=	$\forall x,y\left(\bigvee_{i=1}^{n!/(m!(n-m)!)}\left(\bigwedge_{j=1}^{n} P_{ij}(x,T_j,y)\right)\right)$
(14)	$\zeta(\text{sod})$	:=	$\forall x,y\left(\bigvee_{i=1}^{n!/(m!(n-m)!)}\left(\bigwedge_{j=1}^{n} P_{ij}(x,y,A_j)\right)\right)$
(13′)	$\zeta(\text{cw1})$	:=	$(\neg P(S,T_1,A) \wedge P(S,T_2,A))$
		$\vee$	$(P(S,T_1,A) \wedge \neg P(S,T_2,A))$
(14′)	$\zeta(\text{sod1})$	:=	$(\neg P(S,T,A_1) \wedge \neg P(S,T,A_2) \wedge P(S,T,A_3))$
		$\vee$	$(\neg P(S,T,A_1) \wedge P(S,T,A_2) \wedge \neg P(S,T,A_3))$
		$\vee$	$(P(S,T,A_1) \wedge \neg P(S,T,A_2) \wedge \neg P(S,T,A_3))$

図 3.1 アクセス制御ポリシー変換写像 ζ の定義

3.3.2　アクセス制御ポリシーの形式化

ここでは，アクセス制御のポリシーの集合を第一階述語論理式に変換することにより，矛盾するポリシーや冗長なポリシーの検出を行う手法について論じる．ここで，本章で使用する記号について定義しておく．以下に示すように，アクセス制御ポリシーの集合を $\mathcal{P}$，第一階述語論理式の集合を $\mathcal{L}$ とし，$\mathcal{P}$ から $\mathcal{L}$ への変換写像を ζ と定義する．

$$\zeta : \mathcal{P} \rightarrow \mathcal{L}$$

また，個々のアクセス制御ポリシー，即ち，集合 $\mathcal{P}$ の要素を r としたとき，$r \in \mathcal{P}$ を変換写像 ζ により変換したものを，$\zeta(r) \in \mathcal{L}$ と定義する．

第 3.3.3 項から第 3.3.9 項において，本章で前提とするアクセス制御ポリシー $\mathcal{P}$ とその変換写像 ζ について順に定義する．先に変換写像 ζ の一覧を図 3.1 に示しておく．

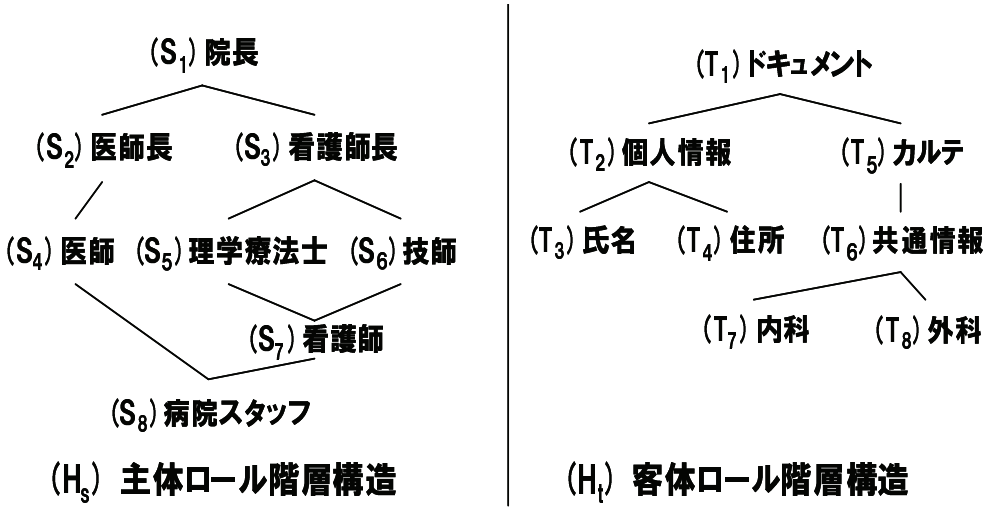

図 3.2 ロール階層構造の例

3.3.3 ロールの形式化

最初にロールについて定義する．RBAC モデルが米国で標準化されているように，アクセス制御ポリシーをロールの概念を用いて定義することは一般的となっている．標準 RBAC モデルの中では，ロールの階層構造は主体のみを前提としているが，これを客体に拡張することは自然な拡張である．木構造で表される半順序関係を持つロールの集合を，本論文では，**ロール階層構造**と呼ぶ．個々の具体的な主体や客体はロールと関係づけられて定義されることになる．特に，ロール階層構造のうち，主体に関係づけられているものを，**主体ロール階層構造** (**SRH:Subject Role Hierarchy**) と呼ぶ．同様に，客体に関係づけられているものを，**客体ロール階層構造** (**TRH:Target Role Hierarchy**) と呼ぶ．これらの階層構造の例を図 3.2 に示す．図 3.2 の左側が，主体ロール階層構造の例であり，最も上位に “(S_1) 院長”が定義され，その下位に，“(S_2) 医師長”と “(S_3) 看護師長”が，そして最下位に “(S_8) 病院スタッフ”が位置づけられる関係を示している．同様に，図 3.2 の右側が，客体ロール階層構造の例であり，最も上位に “(T_1) ドキュメント”が定義され，その下位に，“(T_2) 個人情報”と “(T_5) カルテ”が，そして最下位に “(T_3) 氏名”，“(T_4) 住所”，“(T_7) 内科”，“(T_8) 外科”が位置づけられる関係を示している．

アクセス制御ポリシー自身に加え，ロール階層構造が原因の矛盾を検出するためには，これらの階層構造も論理式に変換する必要がある．階層構造自身はアクセス制御ポリ

シーではないが，便宜上 $\mathcal{H} \in \mathcal{P}$ と定義することとする．そして，写像 $\zeta(\mathcal{H})$ は図 3.1 (1) に示すように定義される．ここで $\mathcal{H}$ は階層構造を意味し，$i \geq_{\mathcal{H}} j$ は階層構造 $\mathcal{H}$ において，i が j より上位であることを意味する．この定義に従えば，例えば，図 3.2 の左側に示される階層構造は，次のように記述されることになる．

$$\begin{aligned}\zeta(\mathcal{H}_s) \quad := \{ \quad & H_{\mathcal{H}_s}(S_1, S_2), H_{\mathcal{H}_s}(S_1, S_3), \\ & H_{\mathcal{H}_s}(S_2, S_4), H_{\mathcal{H}_s}(S_3, S_5), \\ & H_{\mathcal{H}_s}(S_3, S_6), H_{\mathcal{H}_s}(S_5, S_7), \\ & H_{\mathcal{H}_s}(S_6, S_7), H_{\mathcal{H}_s}(S_4, S_8), \\ & H_{\mathcal{H}_s}(S_7, S_8)\}\end{aligned}$$

ここで，$H_{\mathcal{H}}(i, j)$ は 階層構造 $\mathcal{H}$ において，i が j の直近の親ノードであることを意味する．

3.3.4 認可ポリシーの形式化

認可 (許可) ポリシー (Auth+) は，動作 A_1 の実行を主体ロール S_1 が，客体ロール T_1 に対して許可されていることを定義する．同様に認可 (禁止) ポリシー (Auth-) は，動作 A_1 の実行を主体ロール S_1 が，客体ロール T_1 に対して禁止されていることを定義する．例えば，XACML や Ponder では，認可ポリシーは，次のように記載すると定義されている．ただし簡略化のため XACML の記載は正式なものから一部省略している．

XACML:

```
<Rule RuleId="1"
  Effect="Permit|Deny">
  <Subject> S1 </Subject>
  <Resource> T1 </Resource>
  <Action> A1 </Action>
</Rule>
```

Ponder:

```
type auth+|- RuleID_1
  (subject S1, target T1) {
  action A1;
}
```

本論文では，認可ポリシーをそれぞれ次のように定義する．

認可 (許可) ポリシー：$\mathtt{Auth+}(S_1, T_1, A_1) \in \mathcal{P}$.

認可 (禁止) ポリシー：$\mathtt{Auth-}(S_1, T_1, A_1) \in \mathcal{P}$.

認可ポリシーを論理式に変換するための写像 ζ は図 3.1 の (2) 及び (3) に示すように定義する．図 3.1 において，記号 P は 3 つの変数，主体ロール，客体ロール及び動作を引数にもつ第一階述語論理の記号であり，**述語記号**と呼ぶ．述語記号 P により形成される論理式 $P(S_1, T_1, A_1)$ の意味は次のようになる．

- 主体ロール S_1 は，客体ロール T_1 に対して，動作 A_1 を実行することを許可する．

ここで E_x は任意のイベントを意味する．そのため，図 3.1 の (2) 及び (3) は正確には次の意味となる．

(2) 任意のイベント E_x について，S_1 は T_1 に対して，動作 A_1 を許可する．
(3) 任意のイベント E_x について，S_1 は T_1 に対して，動作 A_1 を禁止する．

3.3.5　強制ポリシーの形式化

XACML では，その規程文書の中で強制ポリシーに関して言及しているが，具体的な記述方法までは定義していない．一方で Ponder の中では，強制ポリシーは，イベント，条件，動作の三つ組みルールとして，義務と制約の 2 種類があると規定されている．本章では，それぞれ，**強制 (義務) ポリシー**，**強制 (制約) ポリシー**と呼ぶ．

強制 (義務) ポリシー (Obli+) は，あるイベント E_1 が発生したときに，主体ロール S_1 が，客体ロール T_1 に対して実行しなければならない動作 A_1 を定義したアクセス制御ポリシーである．また，強制 (制約) ポリシー (Obli-) は，あるイベント E_1 が発生したとき

に，主体ロール S_1 が，客体ロール T_1 に対して実行してはいけない動作 A_1 を定義したアクセス制御ポリシーである．

Ponder では，強制 (義務) ポリシーのことを *obligation policy*，強制 (制約) ポリシーのことを *refrain policy* と呼び，それぞれ以下のように記述される．

Ponder:

```
type oblig RuleID_2
  (subject S1, target T1) {
  on E1;
  do A1;
}
type refrain RuleID_3
  (subject S1, target T1) {
  on E1;
  do A1;
}
```

本論文では，強制ポリシーをそれぞれ次のように定義する．

$$\text{強制 (義務) ポリシー：}\texttt{Obli+}(E_1,\ S_1,\ T_1,\ A_1) \in \mathcal{P}.$$

$$\text{強制 (制約) ポリシー：}\texttt{Obli-}(E_1,\ S_1,\ T_1,\ A_1) \in \mathcal{P}.$$

また，強制ポリシーの変換写像 ζ は図 3.1 の (4), (5) のように定義される．

図 3.1 において，記号 O は 3 つの変数，主体ロール，客体ロール及び動作を引数にもつ述語記号であり，述語記号 O により形成される論理式 $O(S_1, T_1, A_1)$ の意味は次のようになる．

- 主体ロール S_1 は，客体ロール T_1 に対して動作 A_1 を実行しなければならない．

同様に記号 R は 3 つの変数，主体ロール，客体ロール及び動作を引数にもつ述語記号であり，述語記号 R により形成される論理式 $R(S_1, T_1, A_1)$ の意味は次のようになる．

- 主体ロール S_1 は，客体ロール T_1 に対して動作 A_1 を実行してはいけない．

つまり，図 3.1 の (4), (5) はそれぞれ次のような意味になる．

(4) あるイベント E_1 が発生したならば，主体ロール S_1 は，客体ロール T_1 に対して動作 A_1 を実行しなければならない．
(5) あるイベント E_1 が発生したならば，主体ロール S_1 は，客体ロール T_1 に対して動作 A_1 を実行してはいけない．

次に，認可ポリシーと強制ポリシーの具体例 r1〜r4 について簡単に述べる．

$$\begin{aligned}
\text{r1} &: \texttt{Auth+}(S_8, T_5, A_7)\\
\text{r2} &: \texttt{Auth−}(S_2, T_5, A_7)\\
\text{r3} &: \texttt{Obli+}(E_1, S_3, T_2, A_8)\\
\text{r4} &: \texttt{Obli−}(E_2, S_2, T_2, A_7)
\end{aligned}$$

ここで，S_2-S_8, T_2-T_5 と A_7-A_8 は，それぞれ，図 3.2 の主体の名称，客体の名称，及び図 3.3 の動作の名称に対応している．

これまで述べてきたように，認可 (許可) ポリシー r1 は，主体 S_8 が，客体 T_5 に対して，動作 A_7 を実行することを許可する．つまり，図 3.2 及び図 3.3 に照らし合わせると，“病院スタッフ”は，“カルテ”に対して “閲覧”という動作を実行することが可能という意味である．

同様に，認可 (禁止) ポリシー r2 は，主体 S_2 が，客体 T_5 に対して，動作 A_7 を実行することを禁止する．つまり，図 3.2 及び図 3.3 に照らし合わせると，“医師長”は，“カルテ”に対して “閲覧”という動作を実行することが禁止されるという意味である．

次に強制ポリシーについて述べる．強制ポリシーは，一般的には，ある条件を満たしたときに，実施しなければならない作業，あるいは実施してはならない作業を定義するポリシーとして利用される．具体的には，強制 (義務) ポリシー r3 は，もしイベント E_1 が発生した場合には，主体 S_3 は客体 T_2 に対して，動作 A_8 を実行しなければならない．つまり，図 3.2 及び図 3.3 に照らし合わせると，もしイベント E_1（例えば，住所変更など）が発生した場合には，“看護師長”は，“個人情報”に対して，“書込”という動作を実行しなければならないという意味である．

同様に，強制 (制約) ポリシー r4 は，もしイベント E_2 が発生した場合には，主体 S_2 は客体 T_2 に対して，動作 A_7 を実行してはいけない．つまり，図 3.2 及び図 3.3 に照らし合わせると，もしイベント E_2（例えば，毎週月曜日など）が発生した場合には，“医師長”

は，“個人情報”に対して，“書込”という動作を実行してはいけないという意味である．

3.3.6　公理

認可ポリシーと強制ポリシーの変換には，述語記号 P, O, R の関係を定義する 2 つの公理が必要となる．一つは，強制 (義務) ポリシーは認可 (許可) ポリシーが定義されていることが前提となるということ．そして，強制 (義務) ポリシーと強制 (制約) ポリシーは矛盾するということである．これらの公理は図 3.1 の (6)，(7) に定義するとおりである．Ax1 は認可 (許可) ポリシーと強制 (義務) ポリシーの両方を含んだ矛盾を検出するために必要となる．また，Ax2 は，強制 (義務) ポリシーと強制 (制約) ポリシーの矛盾を検出するためにそれぞれ必要となる．詳細については後述する．

3.3.7　継承ポリシーの形式化

これまで論じてきたように，アクセス制御ポリシーの主体と客体はロールを用いて定義されるが，そのロールは第 3.3.3 項で述べたように半順序関係があることを前提とする．継承ポリシーとは，ロールの順序関係に従って認可ポリシーがどのように継承されるかを定義するポリシーのことである．

標準的な RBAC モデルや Ponder では，継承の向きはあらかじめ限定されている．しかし，それでは汎用性が低いため，本章では，継承ポリシーは，ロールの構造ごとに，任意の継承の方向を定義できるように汎用性をもたせることとする．継承ポリシーは次のように定義する．

$$\mathtt{prop}(\mathtt{Auth}+|-, \mathtt{SRH}|\mathtt{TRH}, \mathtt{Up}|\mathtt{Down}) \in \mathcal{P}.$$

ここで，SRH と TRH はそれぞれ，継承ポリシーが適用される主体ロールと客体ロールの構造を示している．また，Up と Down は，それぞれロールの構造に従ってどの方向にアクセス制御ポリシーが継承されるかを示している．Up は，半順序関係で最も下位に位置するロールから，上方向にアクセス制御ポリシーが継承されることを意味する．同様に，Down は，半順序関係で最も上位に位置するロールから，下方向にアクセス制御ポリシーが継承されることを意味する．

即ち，全ての継承ポリシーを書き出すと，次に示す 8 種類の継承ポリシーが存在することになる．

$$
\begin{array}{lcl}
\text{prop1} & : & \texttt{prop}(\texttt{Auth+}, \mathcal{H} \in \texttt{SRH}, \texttt{UP}) \\
\text{prop2} & : & \texttt{prop}(\texttt{Auth-}, \mathcal{H} \in \texttt{SRH}, \texttt{Down}) \\
\text{prop3} & : & \texttt{prop}(\texttt{Auth+}, \mathcal{H} \in \texttt{SRH}, \texttt{Down}) \\
\text{prop4} & : & \texttt{prop}(\texttt{Auth-}, \mathcal{H} \in \texttt{SRH}, \texttt{UP}) \\
\text{prop5} & : & \texttt{prop}(\texttt{Auth+}, \mathcal{H} \in \texttt{TRH}, \texttt{UP}) \\
\text{prop6} & : & \texttt{prop}(\texttt{Auth-}, \mathcal{H} \in \texttt{TRH}, \texttt{Down}) \\
\text{prop7} & : & \texttt{prop}(\texttt{Auth+}, \mathcal{H} \in \texttt{TRH}, \texttt{Down}) \\
\text{prop8} & : & \texttt{prop}(\texttt{Auth-}, \mathcal{H} \in \texttt{TRH}, \texttt{UP})
\end{array}
$$

これらの継承ポリシーは必ずしも全てが同時に使用されるとは限らない．一つの継承ポリシーも使用されない場合も想定されるし，複数の継承ポリシーが同時に使用される場合も想定される．どの継承ポリシーが使用されるかは，アクセス制御の対象となるシステムの特性や管理者の判断により決定される．

ここで定義した 8 種類の継承ポリシーは，図 3.1 の (8)〜(11) に示すように 4 種類の論理式にそれぞれ変換される．論理式 $(A \wedge B) \rightarrow C$ と $(\neg C \wedge B) \rightarrow \neg A$ が同値であることと同様に，prop1 と prop2，prop3 と prop4，prop5 と prop6，prop7 と prop8 がそれぞれ同一の論理式に変換されることとなる．

ここで，継承ポリシーの具体例 pr1 について述べる．

$$\text{pr1} : \texttt{prop}(\texttt{Auth-}, \mathcal{H}_{\texttt{s}}, \texttt{DOWN})$$

ただし，$\mathcal{H}_s$ は図 3.2 の左側の主体ロール階層構造を示しているとする. 継承ポリシー pr1 は，認可 (禁止) ポリシーが，階層構造 $\mathcal{H}_s$ に従って，下向きに継承されることを定義している．つまり，前述の認可 (禁止) ポリシー r2 から，次の 2 つの認可 (禁止) ポリシーが継承ポリシーにより暗黙的に定義されることを意味する．

$$
\begin{array}{lcl}
\text{r2\_1} & : & \texttt{Auth}-(S_4, T_5, A_7) \\
\text{r2\_2} & : & \texttt{Auth}-(S_8, T_5, A_7)
\end{array}
$$

すると，r2_2 は r1 と矛盾の関係にあることがわかる．このように，継承ポリシーは，暗黙的に矛盾するポリシーの組み合わせを発生させる可能性があることに注意が必要である．

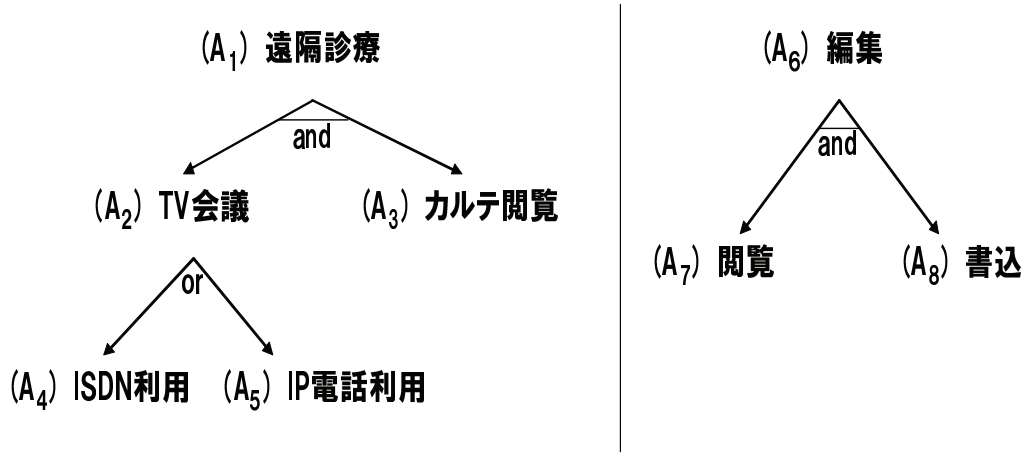

図 3.3 合成動作の例

3.3.8　合成動作ポリシーの形式化

主体や客体のロールと同様に，動作と動作の間にも何らかの関係があることを考えることは自然な拡張である．本章では，動作と動作の関係を定義したポリシーを合成動作ポリシーと呼ぶ．合成動作の例を，図 3.3 に示す．

例えば，図 3.3 に示すように，3 つの動作，A_1, A_2, A_3 に対して，次のような合成動作ポリシーを考えることができる．

$$A_1 = A_2 \wedge A_3$$

これは，一つの動作 A_1 を実行することと，2 つの動作 A_2，A_3 を同時に実行することは等価であるという意味である．つまり，認可 (許可) ポリシー

$$\text{r5} \quad : \quad \texttt{Auth+}(S_1, T_1, A_1)$$

は，次の二つの認可 (許可) ポリシーと同じ意味を持つことになる．

$$\begin{aligned} \text{r6} \quad &: \quad \texttt{Auth+}(S_1, T_1, A_2) \\ \text{r7} \quad &: \quad \texttt{Auth+}(S_1, T_1, A_3) \end{aligned}$$

合成動作ポリシーは，n 個の動作 $A_1, \cdots, A_n$ の組み合わせにより次のように定義する．

$$A_1 = \Gamma(A_2, \cdots, A_n)$$

ただし，Γ は，$A_2, \cdots, A_n$ を，論理結合子 $\wedge, \vee, \neg$ で任意に結合した論理式で構成される関数とする．この合成動作ポリシーの変換は，図 3.1(12) のように定義される．

次に図 3.3 を用いて合成動作ポリシーの具体例について述べる．例えば図 3.3 の左側は，次に示すような依存関係があることを示している．

ac1　：　遠隔診療 = TV 会議 $\wedge$ カルテの閲覧
ac2　：　TV 会議 = ISDN 利用 $\vee$ IP 電話利用

ここで，合成動作ポリシー ac1 は，“遠隔診療”の実行は，“TV 会議”と “カルテの閲覧”の 2 つの動作を許可されている場合にのみ，許可されることを定義している．

同様に，合成動作ポリシー ac2 は，“TV 会議”の実行は，“ISDN 利用”か “IP 電話利用”のいずれかの動作を許可されている場合にのみ，許可されることを定義している．

これらの合成ポリシーもまた継承ポリシーと同様に，暗黙的な矛盾を生じる可能性がある．例えば，以下のような許可ポリシーがあるとする．

r8　：　Auth+(S_4, T_2, A_1)
r9　：　Auth−(S_4, T_2, A_2)
r10　：　Auth−(S_4, T_2, A_3)

アクセス制御ポリシー r8 は “医師”が “遠隔診療”を実行することを許可している．一方で，r9 と r10 では，“遠隔診療”を実行するために必要なる，“TV 会議”と “カルテ閲覧”の実行を禁止している．つまり，これら 3 つのポリシーは矛盾している．このような矛盾を本論文では，制約矛盾と呼ぶが，詳細については後述する．

3.3.9　制約ポリシーの形式化

最後に，チャイニーズウォールポリシー[6] と職務分掌ポリシー[10] の定義と形式化について述べる．チャイニーズウォールポリシーと職務分掌ポリシーは，それぞれ，客体ロールと，動作に関して数的な制約条件を定義するアクセス制御ポリシーである．

チャイニーズウォールポリシーは，主体ロールが，ある動作を実行することのできる客体ロールの数に制約を科すポリシーである．一方で，職務分掌ポリシーは，主体ロールが，客体ロールに対して実行することのできる動作の数に制約を科すポリシーである．

チャイニーズウォールポリシーも職務分掌ポリシーも，制約の有無がシステムの状態に応じて動的に変更される場合と，システムの状態に無関係に静的に定義される場合が考えられる．本章では，静的に定義されるポリシーのみを対象とする．

チャイニーズウォールポリシーと職務分掌ポリシーは，それぞれの次のように定義する．

$$\begin{aligned} \text{cw} &: \mathtt{CW}(all, \{T_1, \cdots, T_n\}_m, all) \in \mathcal{P} \\ \text{sod} &: \mathtt{SoD}(all, all, \{A_1, \cdots, A_n\}_m) \in \mathcal{P} \end{aligned}$$

ここで，m $(0 < m < n)$ は，客体ロールと動作に関する制約数を意味する．つまり，ポリシー cw は，

- 任意の主体ロールは，任意の動作を，n 個の客体ロール $\{T_1, \cdots, T_n\}$ のうち，最大で m 個に対して，実行することが許可される．

という意味である．そして，ポリシー sod は，

- 任意の主体ロールは，任意の客体ロールに対して，n 個の動作 $\{A_1, \cdots, A_n\}$ のうち，最大で m 個の動作を実施することを許可される．

という意味である．

チャイニーズウォールポリシーと職務分掌ポリシーはそれぞれ図 3.1 の (13), (14) のように変換写像が定義される．ここで，$P_{ij}(\cdot) = P(\cdot)$ or $\neg P(\cdot)$ である．$P(\cdot)$ と $\neg P(\cdot)$ は，m 個の $P(\cdot)$ と，$n - m$ 個の $\neg P(\cdot)$ が過不足無く記載されるように定義される必要があることに注意する．

例えば，チャイニーズウォールポリシーに関して，特定の主体ロールや，動作に関して制約を定義する必要がある場合には，変数 x,y の代わりに，図 3.1 の (13') に示すように，具体的な値を当てはめて定義すればよい．職務分掌に関しても同様であり，その例は，図 3.1 の (14') のようになる．

次に，チャイニーズウォールポリシーと職務分掌ポリシーの具体例について述べる．

チャイニーズウォールポリシーは，インサイダー取引などの不正を防止するために，金融機関において多く導入されているポリシーである．具体的には，競合関係にある企業の財務データに同一のユーザがアクセスすることを防ぐことが目的のアクセス制御ポリシーである．チャイニーズウォールポリシーは次のように記述される．

$$\mathrm{cw1} : \mathtt{CW}(S_8, \{T_2, T_5\}_1, A_7)$$

cw1 は，主体 S_8 は客体 T_2，T_5 のうちどちらか一つに対してのみ，動作 A_7 を実行することを許可することを規定している．これは，明らかに次の 2 つのアクセス制御ポリシーと矛盾関係にある．

$$\begin{aligned} \mathrm{r11} &: \mathtt{Auth+}(S_8, T_2, A_7) \\ \mathrm{r12} &: \mathtt{Auth+}(S_8, T_5, A_7) \end{aligned}$$

同様に，職務分掌ポリシーとは，動作に関して制約を加えたポリシーであり，例えば，次のように記述される．

$$\mathrm{sod1} : \mathtt{SoD}(S_8, T_2, \{A_7, A_8, A_9\}_2)$$

sod1 は，主体 S_8 は，客体 T_2 に対して，3 つの動作 A_7，A_8，A_9 のうち，高々 2 つの動作のみ実行を許可されるという意味である．職務分掌ポリシーも，その定義から明らかなように，チャイニーズウォールポリシーと同様の矛盾を生じる可能性がある．

3.4 タブロー法

3.4.1 タブロー法の概要

本項では，矛盾ポリシーと冗長ポリシーの検出を可能とする自由変数タブロー法について，その概略を述べる．

主体や客体の構造に準じてポリシーが継承するという概念は一般的であり，ポリシーを定義する管理者にとっても，アクセス制御ポリシーの記述の手間を簡略化するという観点から必要不可欠である．動作の構造やチャイニーズウォールポリシー，職務分掌ポリシーも同様である．しかしながら，前節で見たように，これらのポリシーは暗黙的に他のポリシーとの制約関係を生じさせるため，矛盾するポリシーや使用されることのない冗長なポリシーを定義する可能性があり，管理が複雑になるという新たな問題が発生する．

セキュリティとシステムの性能を確保する観点から，矛盾するポリシーや冗長なポリシーがアクセス制御ポリシーの中に含まれている場合には，それを確実に検出することが管理者に求められる．ところが，次節で述べるように，本章で対象とするアクセス制御モデルには，いくつかの異なる種類の矛盾が生じる可能性がある．単純に考えると，ポリシーの矛盾の種類毎に異なる矛盾検出アルゴリズムを実装するという方法が考えられるが，明らかに効率的な方法とはいえない．さらにシステムの拡張性の観点からも避けるべきである．なぜならば，アクセス制御モデルが拡張され，新たな種類のポリシー矛盾が出現した場合に，迅速に対応することができないからである．クラウドコンピューティングのように様々なシステムが連携することが前提となる近年の環境においては，アクセス制御モデルの変更に迅速に対応できることは重要であり，新たな矛盾が生じる都度，新たな矛盾検出アルゴリズムを実装する方式は現実的とはいえない．

そこで本章では，自由変数タブロー法を用いたアクセス制御ポリシーの整合性検証技術について論じる．自由変数タブロー法は，様々な異なるタイプの矛盾や冗長ポリシーをシンプルで単一のアルゴリズムにより検出することを可能とする．さらに，矛盾を解消するためのヒントとなる情報を推測して管理者に提示することが可能であるという利点もある．

論理式の集合 Γ から C が帰結することを証明すること，即ち，$\Gamma \models C$ を証明することは，集合 $\{\Gamma, \neg C\}$ が矛盾しているということ，即ち $\{\Gamma, \neg C\} \models \bot$ ということと同値である．タブロー法はこの性質を利用して論理式の矛盾検出を可能にするアルゴリズムである．

さらにタブロー法を拡張した自由変数タブロー法を用いることにより，ポリシーの集合が矛盾を含んでいることのみではなく，あるポリシーが，他のポリシーの集合から導出可能であるかどうかも判定することが可能となる．自由変数タブロー法は充足的かつ完全な定理証明手法であり，簡単な演繹手法をも備えている．さらにそのアルゴリズムがシンプルであり，システムへの実装が容易という特徴がある．

自由変数タブロー法を用いて矛盾ポリシーと冗長ポリシーを検出する手順の概略について述べる．

矛盾ポリシーの検出手順

自由変数タブロー法を用いて矛盾ポリシーを検出する手法は，大きく二つの手順からなる．

i) それぞれのポリシー $r_i \in \mathcal{P}$ を論理式 $\zeta(r_i)$ に変換する．ここで，$\mathcal{P}$ はポリシーの集合を意味する．
ii) 次に自由変数タブロー法を集合 $\{\zeta(r_i)\}$ に対して適用し矛盾を検出する．このとき，矛盾の原因となる情報も同時に導出する．

冗長ポリシーの検出手順

自由変数タブロー法を用いて冗長ポリシーを検出する手法は，大きく二つの手順からなる．

i) それぞれのポリシー $r_i \in \mathcal{P}$ と，ポリシーの集合 $\mathcal{P}$ から導出されることを確認したいポリシー r を，論理式 $\{\zeta(r_i)\}$ と $\zeta(r)$ にそれぞれ変換する．
ii) 次に自由変数タブロー法を集合 $\{\{\zeta(r_i)\}, \neg\zeta(r)\}$ に対して適用し，矛盾を検証することによって，ポリシー r が他のポリシーから導出可能か否かを判定する．あるポリシーが他のポリシーから導出可能ということは，一方のポリシーが冗長なポリシーであること意味する．

これらの手順の詳細については，後述する．

上記で述べたように，矛盾ポリシーと冗長ポリシーを自由変数タブロー法を用いて検出可能にするためには，第 3.3.2 項で定義した以下の写像が，アクセス制御ポリシー $\mathcal{P}$ の矛盾と第一階述語論理式 $\mathcal{L}$ の矛盾が同値になるように適切に定義されていればよいことになる．

$$\zeta : \mathcal{P} \rightarrow \mathcal{L}$$

即ち，矛盾するポリシーの集合 $\mathcal{P}' \subset \mathcal{P}$ を写像 ζ によって変換した集合 $\zeta(\mathcal{P}')$ が矛盾している集合となっていればよい．このことを示すためには，最初に矛盾ポリシーと冗長ポリシーを定義する必要がある．これらの定義については，第 3.5 節及び第 3.6 節で述べる．

3.4.2 自動定理証明手法の特徴

変換対象とする論理式として，第一階述語論理式ではなく，様相論理などを利用することも考えられる．それにより，より複雑なアクセス制御ポリシーを対象とすることが可能になる．しかしながら，その反面，論理式の定理証明が非常に複雑になり，実装に適さないという問題が発生する[33]．そこで，本章では，アクセス制御ポリシーの変換写像を第一階述語論理式に制限することとする．

例えば，OTTER,[23),34)] PTTP[43)]，Setheo[29)]や LeanTap[3)]などの自動定理証明手法は，アルゴリズムとして自由変数タブロー法を採用している．これは，自由変数タブロー法のアルゴリズムがシンプルであり，その実装が容易な上に適用範囲が広いという性質に基づいている．また，自由変数タブロー法のアルゴリズムの実装についてもいくつかが公開[2),39)] されており，TPTP[45)] と呼ばれる定理証明のベンチマーキング基準も整備され，それとの比較結果が示されるなど，広く研究されている手法であるということができる．

アクセス制御ポリシーの矛盾を検出するためには，大規模なポリシーを高速に解析でき，かつポリシーの変更にも柔軟に対応可能なアルゴリズムが必要であることから，自由変数タブロー法を適用することとした．

3.4.3 タブロー法によるアクセス制御ポリシーの検証

自由変数タブロー法そのものは，論理学の議論であり，本論文の本質ではないなため，詳細な説明に関しては，M. Fitting の著書[15)]を参照することとする．本項では，ポリシーの解析を理解する上で必要となる点に焦点を絞り，自由変数タブロー法による解析手順を簡単に紹介する．

ここでは，図 3.4 に示す事例を用いて，自由変数タブロー法の解析手順の概要について述べる．図 3.4 は，以下に示す 2 つのアクセス制御ポリシー r13， r14 が矛盾であることを自由変数タブロー法により解析した結果を示している．

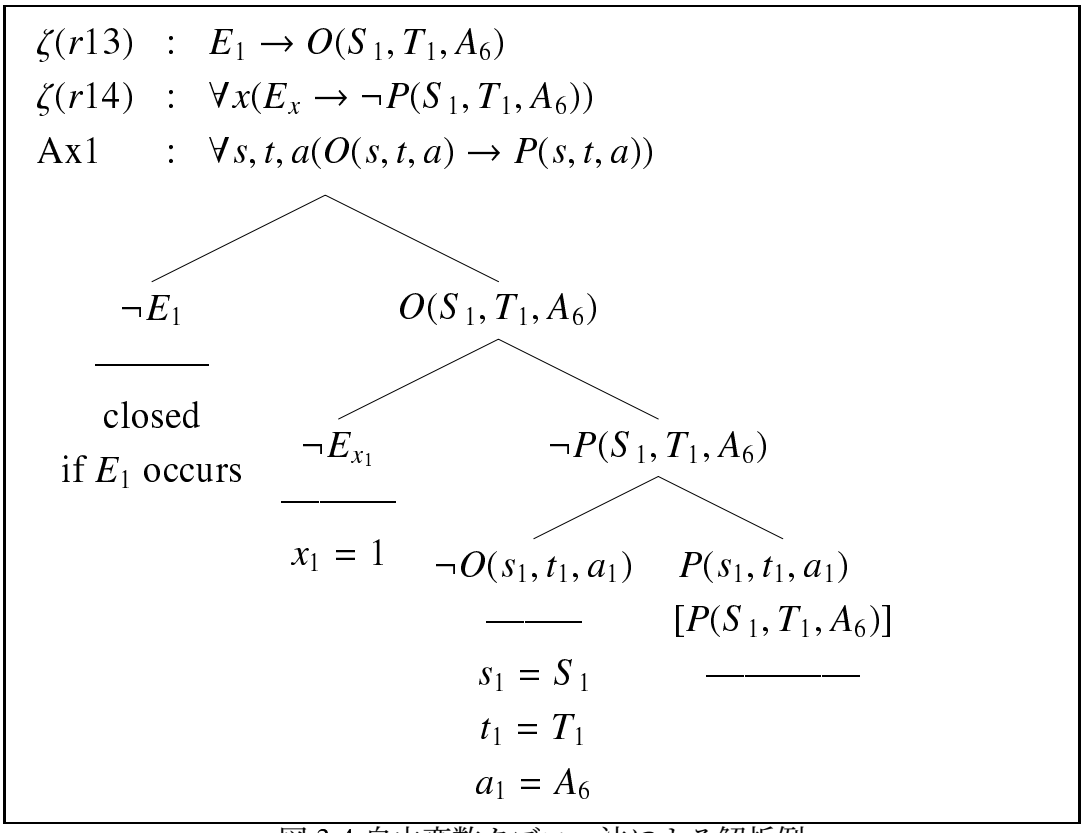

図 3.4 自由変数タブロー法による解析例

$$\text{r13} \quad : \quad \texttt{Obli+}(S_1, T_1, A_6)$$
$$\text{r14} \quad : \quad \texttt{Auth-}(S_1, T_1, A_6)$$

アクセス制御ポリシー r13，r14 はそれぞれ次のような意味であり，日本語の意味において矛盾していることは明らかである．

(r13) “(S_1) 医師長”は，“(T_1) ドキュメント”に対して，“(A_6) 編集”しなければならない
(r14) “(S_1) 医師長”は，“(T_1) ドキュメント”に対して，“(A_6) 編集”することができない

自由変数タブロー法は図 3.4 に示すように，分析対象となる論理式を起点として，下側に木構造に分割をしていきながら，矛盾を判断するという特徴を持つ．ただし，この例では，認可 (禁止) ポリシーと義務 (強制) ポリシーの矛盾を検出するため，それぞれの述語記号 P と O の関係を定義した公理 Ax1 を起点となる論理式に加えている．自由変数タブロー法は分割された経路の中に矛盾する論理式が存在する場合にその経路を閉じる．経路が閉じられたことを図 3.4 では，水平線で示している．経路を閉じるか否かの判定基

準は単純であり，同一経路内に A と $\neg A$ という明らかに矛盾する論理式の組が出現した場合にのみ，経路を閉じる．分析の結果，枝分かれしたすべての経路が閉じられた場合には，最初に起点とした論理式の集合が矛盾していることを証明したことになる．仮に一つでも経路が閉じられない場合には，起点とした論理式の集合が矛盾していないことを証明したことになる．

図 3.4 の解析手続きについてもう少し詳しく説明する．自由変数タブロー法の分析は与えられた集合が矛盾しないという仮定をおいて進めていく．そして，その分析の結果，万が一矛盾が発生する場合には，最初の矛盾がないという仮説が誤っていたということになる．即ち，背理法的に最初の集合に矛盾が含まれていたと判断する．

例えば $A \rightarrow B$ が真であるということは，$\neg A$ が真か B が真であることを意味する．このことは，自由変数タブロー法では，$A \rightarrow B$ という論理式を解析する場合には，$\neg A$ と B の 2 つの枝に分割して解析することを意味する．図 3.4 では，$E_1 \rightarrow O(S_1, T_1, A_6)$ という論理式が，$\neg E_1$ と $O(S_1, T_1, A_6)$ に分割されることになる．つまり，「あるイベント E_1 が発生したならば，“(S_1) 医師長”は，“(T_1) ドキュメント”に対して，“(A_6) 編集”しなければならない」というアクセス制御ポリシーが真になるためには，あるイベント E_1 が発生しない $(\neg E_1)$ か，“(S_1) 医師長”は，“(T_1) ドキュメント”に対して，“(A_6) 編集”しなければならない $(O(S_1, T_1, A_6))$ のいずれかが真である必要がある．

このとき，イベント E_1 が発生することは，アクセス制御ポリシー r13 が適用される前提条件である．従ってイベント E_1 が発生したと仮定すると，一番左側の経路に E_1 と $\neg E_1$ という矛盾する論理式の組み合わせが出現するので，一番左側の経路は閉じられることになる．次に同様に，アクセス制御ポリシー r14 を分割すると，$\neg E_{x_1}$ と $\neg P(S_1, T_1, A_6)$ に分割される．ここで，前述したように，イベント E_1 が発生していることは前提となるため，左から 2 つめの経路は閉じられる．この時点で，一番右側の経路はまだ閉じられていない．

そこで，最後に，Ax1 を $\neg O(s_1, t_1, a_1)$ と $P(s_1, t_1, a_1)$ に分割する．自由変数タブロー法では，全称記号 $(\forall)$ の対象となる任意変数（ここでは，s, t, a）を含む論理式を分割する際には，ある特定の定数（ここでは，s_1, t_1, a_1）を代入して分割する．さらに，その特定の定数がある条件を満足する時に，その経路が閉じる場合には，その条件を記述して経路を閉じる．具体的には，図 3.4 の左から 3 番目の経路において，$s_1 = S_1, t_1 = T_1, a_1 = A_6$ という条件を満たすとき，同一経路上に，$O(S_1, T_1, A_6)$ と $\neg O(S_1, T_1, A_6)$ という矛盾する論理式の組が出現するので経路が閉じられる．そして，この条件 $s_1 = S_1, t_1 = T_1, a_1 = A_6$ は，

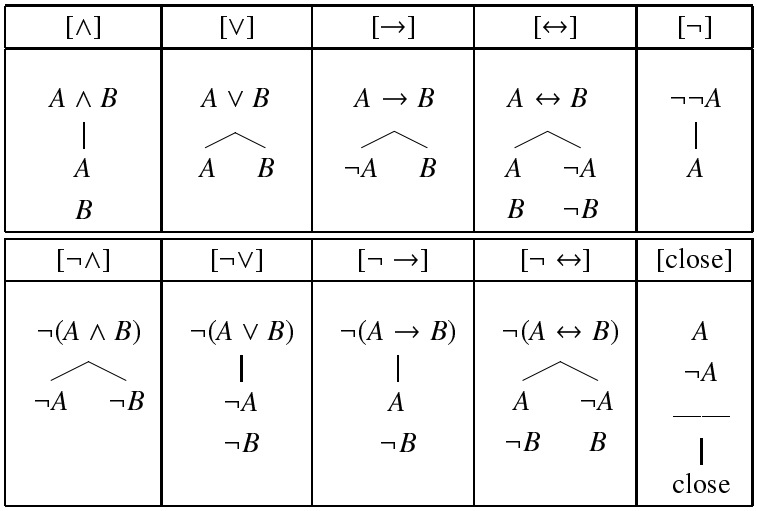

図 3.5 自由変数タブロー法の検証規則

それ以降の分析の前提となる．そのため，図 3.4 の一番右側の経路に出現した $P(s_1, t_1, a_1)$ は，$P(S_1, T_1, A_6)$ に置き換わる．自由変数タブロー法では，解析の途中で定義した条件を代入した場合には，それを大括弧 [] で記述する．すると，一番右側の経路にも，$P(S_1, T_1, A_6)$ と $\neg P(S_1, T_1, A_6)$ という矛盾する論理式の組が出現するので経路が閉じられる．即ち，全ての経路が閉じられたため，アクセス制御ポリシー r13，r14 は矛盾していることが示されたことになる．

ここで，自由変数タブロー法において解析の途中で用いた条件，即ち，$s_1 = S_1, t_1 = T_1, a_1 = A_6$ が矛盾の原因となる情報を提供することになる．図 3.4 の例では，矛盾の原因が，“(S_1) 医師長”，“(T_1) ドキュメント”，“(A_6) 編集”の何れかにあるということを示している．このように，自由変数タブロー法では，矛盾の原因となる情報を解析結果として提供することが可能である．

図 3.4 では，解析の過程において，$A \rightarrow B$ という形式の論理式のみが使用されたが，他の論理式も同様に解析が可能である．第一階述語論理式を解析する際に利用される自由変数タブローの検証規則のすべてを図 3.5 に示しておく．

3.5　アクセス制御ポリシーの矛盾検出

本節では，自由変数タブロー法により，数種類の異なるタイプの矛盾を検出することが可能であることを示す．最初に，第 3.3 節で述べたアクセス制御ポリシーモデルには，単純矛盾，暗黙的な単純矛盾，制約矛盾と呼ばれる 3 種類の矛盾が存在することを述べ，それらを数学的に定義する．そして，定義した矛盾が自由変数タブロー法によってすべて検出可能であることを示す．

3.5.1　単純矛盾

Lupu[31] らは，2 つの認可ポリシー，あるいは，2 つの強制ポリシー，あるいは，1 つの認可ポリシーと 1 つの強制ポリシーがある条件で組み合わされた場合に，矛盾が生じると定義している．具体的には，ポリシーの中で定義された主体と客体と動作が同一となるような {Auth+/Auth-}, {Obli+/Obli-}, {Obli+/Auth-} の 3 種類の組み合わせは，いずれもその意味において矛盾するポリシーの組み合わせとなる．これらの組み合わせは，明示的に定義される場合と，継承ポリシーにより暗黙的に定義される場合がありえるが，ここでは，明示的に定義された場合の矛盾を単純矛盾，暗黙的に定義される矛盾を暗黙的な単純矛盾として区別して定義することとする．本項ではまず，単純矛盾の定義について述べる．

定義 1 (単純矛盾)**.** 任意の定数 C について, 次に示す [Type I]〜[Type III] のペアは，単純矛盾であると定義する.

[Type I] (Auth+/Auth− Conflict)

$$\begin{aligned} \text{r15} &: \texttt{Auth+}(S_C, T_C, A_C) \\ \text{r16} &: \texttt{Auth−}(S_C, T_C, A_C) \end{aligned}$$

[Type II] (Obli+/Obli− Conflict)

$$\begin{aligned} \text{r17} &: \texttt{Obli+}(E_C, S_C, T_C, A_C) \\ \text{r18} &: \texttt{Obli−}(E_C, S_C, T_C, A_C) \end{aligned}$$

[Type III] (Obli+/Auth− Conflict)

$$\begin{aligned} \text{r19} &: \texttt{Obli+}(E_C, S_C, T_C, A_C) \\ \text{r20} &: \texttt{Auth−}(S_C, T_C, A_C) \end{aligned}$$

これら 3 種類の単純矛盾に関して，次の定理が成り立つ.

定理 1 (単純矛盾検出の完全性)**.** 全ての単純矛盾は，自由変数タブロー法により，矛盾であると判定される.

証明. 定理 1 を証明するためには，定義 1 で定義された，3 種類の単純矛盾それぞれについて，自由変数タブロー法を適用し，解析してみればよい．解析するためには，最初に図 3.1 で定義した写像 ζ に従い，それぞれのポリシー r15 〜r20 を変換してから，自由変数タブロー法を適用することになる．その結果を図 3.6〜図 3.8 に示す．図 3.6〜図 3.8 に示されるように，[Type I]，[Type II] 及び [Type III] のすべてについて，自由変数タブロー法による解析の結果，全ての経路が閉じられるため，矛盾であることが証明される．[Type I] に関しては，何か特定のイベントが発生しない限り，矛盾にはならないため，イベント E_C が発生したと仮定している． □

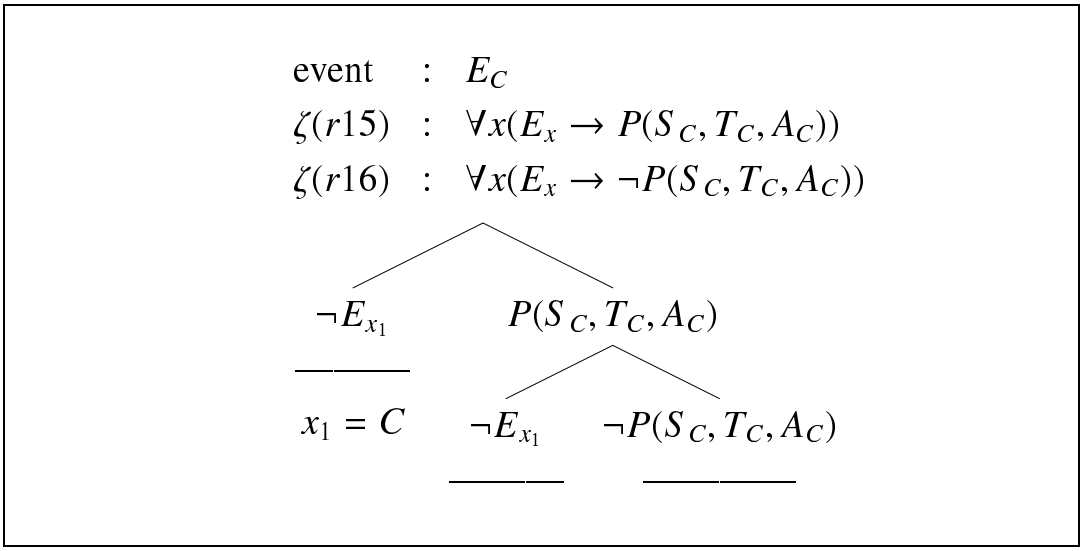

図 3.6 [Type I] 単純矛盾 (Auth+/Auth−) の検出

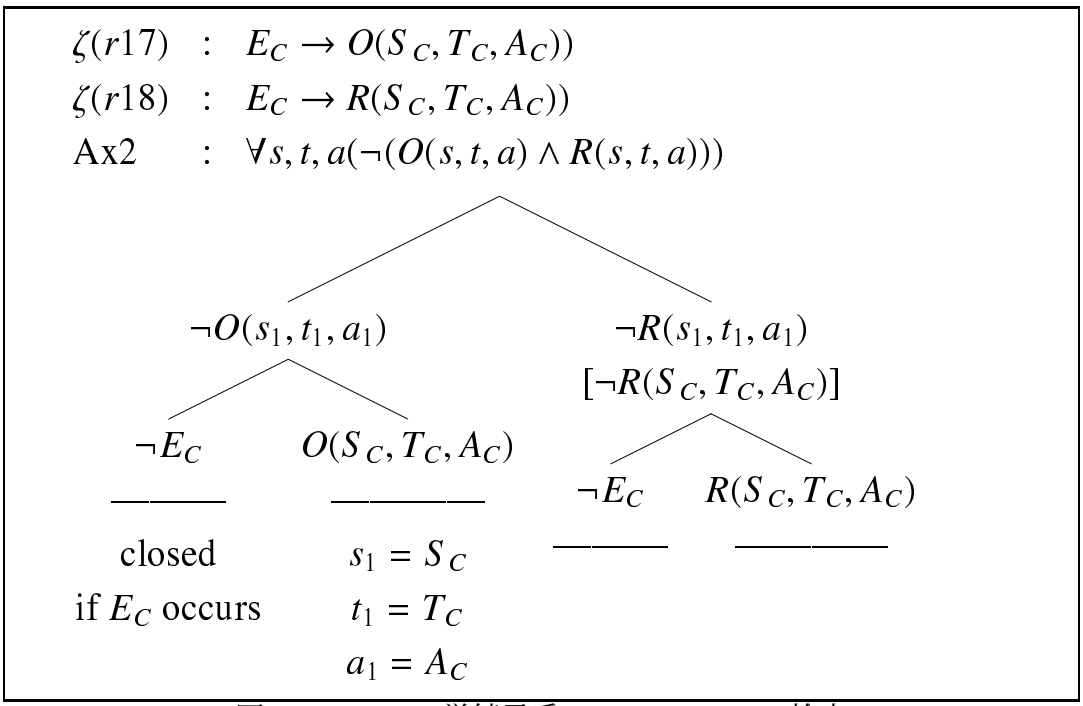

図 3.7 [Type II] 単純矛盾 (Obli+/Obli−) の検出

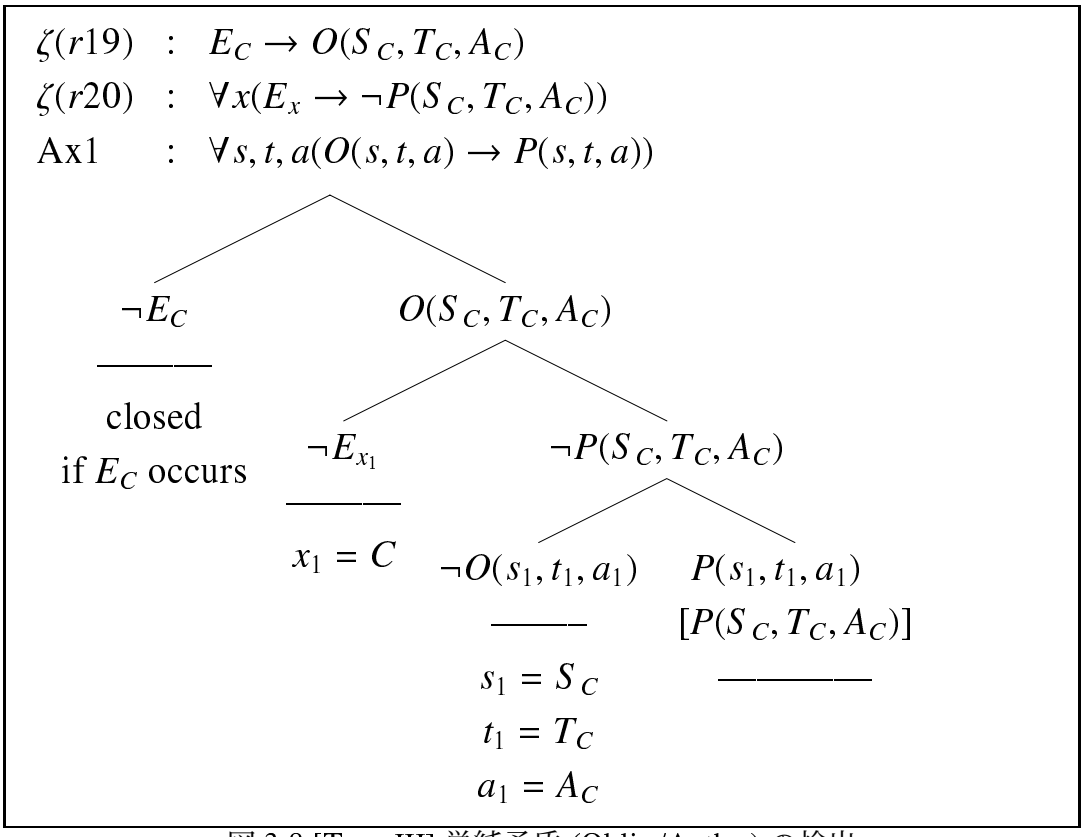

図 3.8 [Type III] 単純矛盾 (Obli+/Auth−) の検出

3.5.2　暗黙的な単純矛盾

本項では，継承ポリシーによって誘発される暗黙的な単純矛盾に関して述べる．

定義 2 (暗黙的な単純矛盾)**.** 任意の 2 つの認可ポリシー

$$\texttt{Auth+}(S_C^+, T_C^+, A_C^+),$$

$$\texttt{Auth−}(S_C^-, T_C^-, A_C^-),$$

に対して，

$$\Omega_+ = \{\texttt{Auth+}(S_{C_i}^+, T_{C_i}^+, A_C^+)\}$$

$$\Omega_- = \{\texttt{Auth−}(S_{C_i}^-, T_{C_i}^-, A_C^-)\}$$

を，それぞれ継承ポリシーにより導出される認可ポリシーの集合であるとする．このとき，集合 $\Omega_+ \cup \Omega_-$ 内，または，集合 $\Omega_+ \cup \Omega_-$ 内のポリシーとそれとは別に定義されてい

る認可ポリシーまたは強制ポリシーの間に，単純矛盾が発生する場合に，その矛盾を暗黙的な単純矛盾と定義する．

定理 2 (暗黙的な単純矛盾の検出)**.** 全ての暗黙的な単純矛盾は自由変数タブロー法により検出可能である．

この定理 2 を証明する前に，第 3.3 節で述べた r1，r2，pr1 が矛盾であることを自由変数タブロー法により判定できることを示す．自由変数タブロー法の適用結果は図 3.9 に示すとおりである．前節同様に矛盾は何らかのイベントが発生したときのみ検出されるので，図 3.9 ではイベント E_1 が発生したと仮定して分析をしている．

分析結果を見てみると，各変数が値 $\{S_8,\ T_5,\ S_7,\ A_7\}$，値 $\{S_4,\ T_5,\ S_2,\ A_7\}$ をとったときに矛盾が発生していることがわかる．このことから，$\{S_2, S_4, S_8\}$ という継承が原因で矛盾が発生していることを推定することができる．

証明. 定理 1 において，既に全ての単純矛盾が自由変数タブロー法により検出されることは示した．つまり，定理 2 を証明するためには，全ての認可ポリシーが，定義 2 で示した集合 $\Omega_+ \cup \Omega_-$ の中に包含されていることを証明すればよい．しかしながら，このことは，定義より明らかである． □

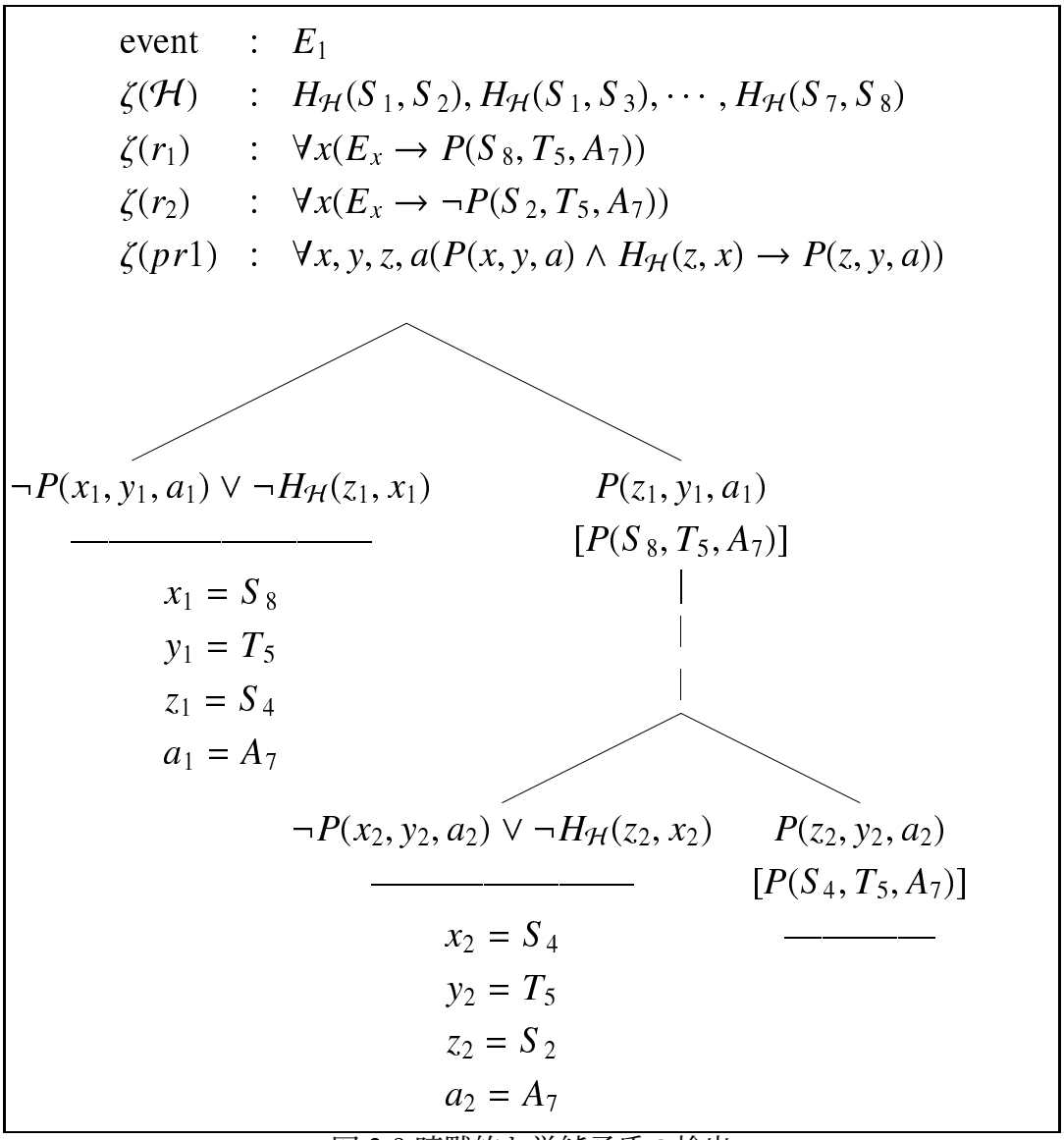

図 3.9 暗黙的な単純矛盾の検出

3.5.3 制約矛盾

本項では，合成動作，チャイニーズウォールポリシー，職務分掌ポリシーにより引き起こされる制約矛盾に関して定義し，それらが，自由変数タブロー法によって検出可能であることを証明する．

合成動作による制約矛盾

合成動作による制約矛盾は次のように定義される．

定義 3 (合成動作による制約矛盾)**.** 合成動作により発生する制約矛盾には，3 種類の基本形がある．それらは次に示す組み合わせ [Type IV]〜[Type VI] で定義される．例えば，[Type IV] に関しては，4 つのポリシー ac1, r21, r22, r23 が全て揃った場合に，矛盾となることを意味する．

[Type IV]

ac1 : $A_1 = A_2 \vee A_3$
r21 : $\texttt{Auth+}(S_C, T_C, A_1)$
r22 : $\texttt{Auth−}(S_C, T_C, A_2)$
r23 : $\texttt{Auth−}(S_C, T_C, A_3)$

[Type V]

ac2 : $A_1 = A_2 \wedge A_3$
r24 : $\texttt{Auth+}(S_C, T_C, A_1)$
r25 : $\texttt{Auth−}(S_C, T_C, A_2)$ or
$\texttt{Auth−}(S_C, T_C, A_3)$

[Type VI]

ac3 : $A_1 = \neg A_2$
r26 : $\texttt{Auth+}(S_C, T_C, A_1)$
r27 : $\texttt{Auth+}(S_C, T_C, A_2)$

ここで，C は任意の定数を意味する．また，認可ポリシー Auth+/-は，明示的に定義されている場合のみではなく，継承ポリシーにより暗黙的に導出される場合も含むものとする．これら以外の合成動作が原因の矛盾ポリシーは，この 3 種類の基本形の組み合わせにより演繹的に定義される．

定理 3 (制約矛盾 (合成動作) の検出)**.** 全ての合成動作により発生する制約矛盾は，自由変数タブロー法により検出可能である．

証明. 合成動作が原因の全ての制約ポリシーは，[Type IV], [Type V] 及び [Type VI] の組み合わせにより定義される．そこで，基本形となるこれら 3 種類の矛盾を自由変数タブロー法により検出することが示されればよい．[Type IV], [Type V] 及び [Type VI] のポリ

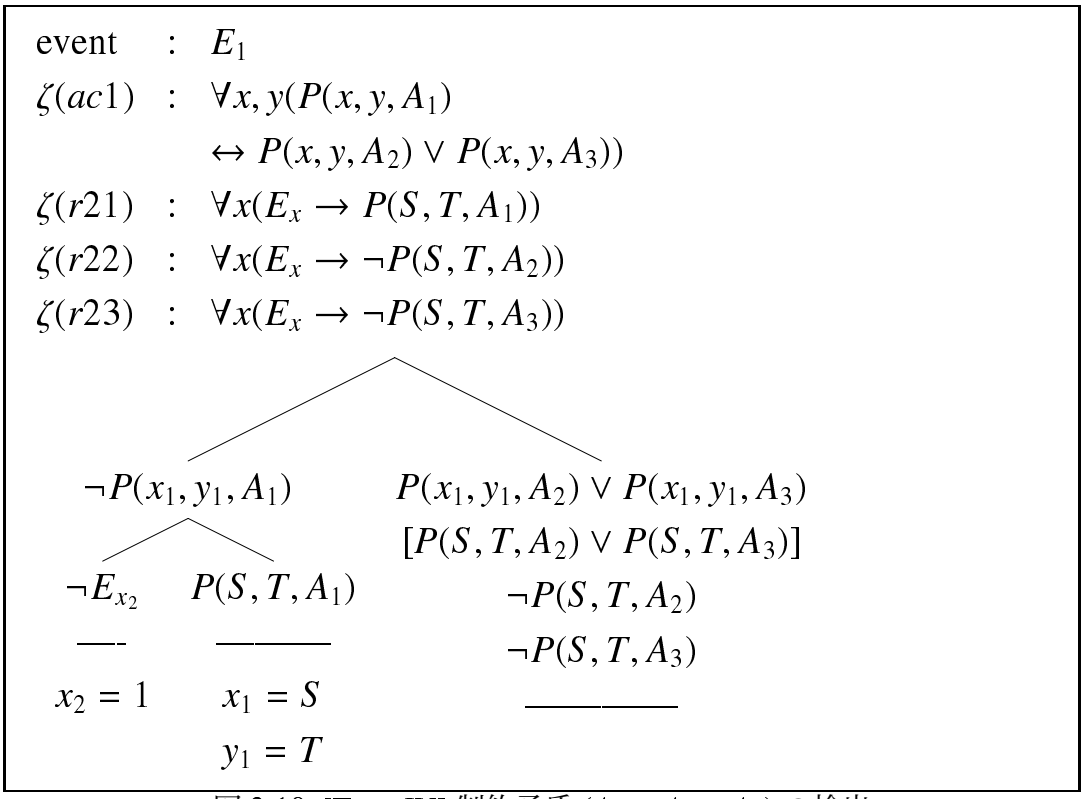

図 3.10 [Type IV] 制約矛盾 ($A_1 = A_2 \vee A_3$) の検出

シーをタブロー法を用いて解析した結果を，図 3.10〜図 3.12 に示す．これらの図に示すように，すべてにおいて矛盾であることが判定されている． □

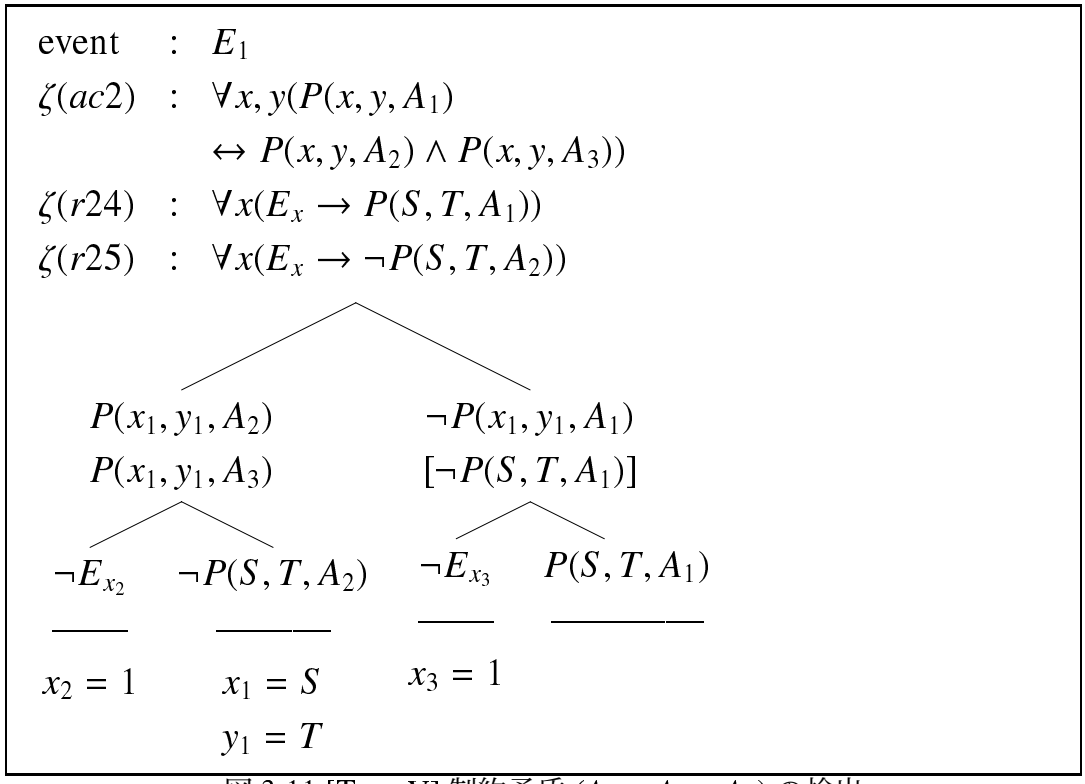

図 3.11 [Type V] 制約矛盾 ($A_1 = A_2 \wedge A_3$) の検出

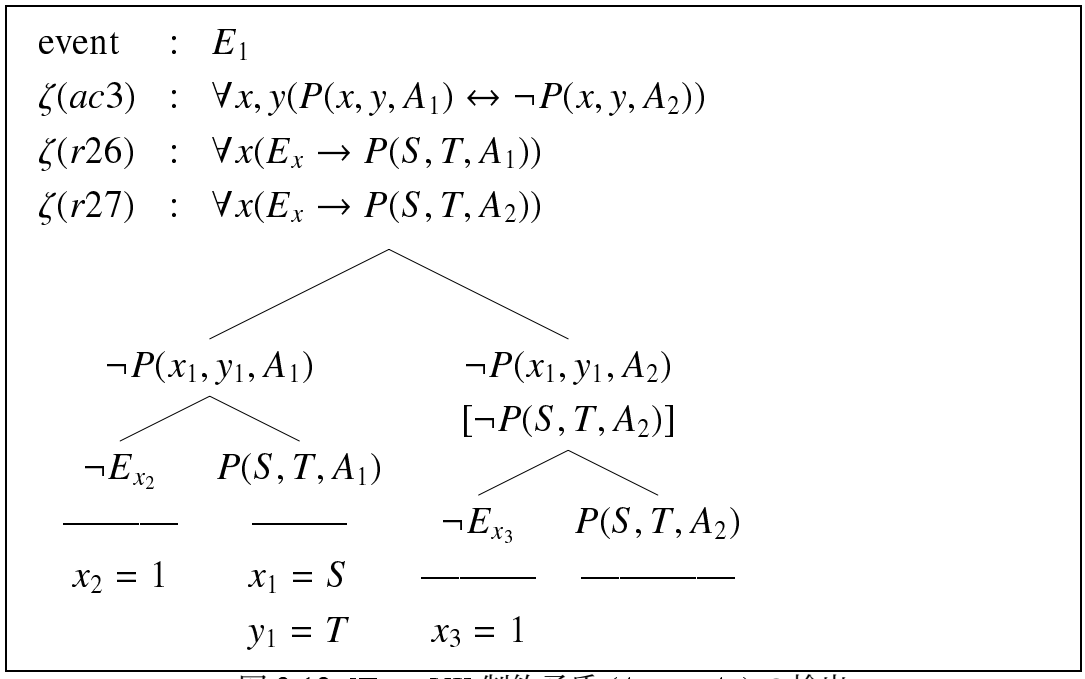

図 3.12 [Type VI] 制約矛盾 ($A_1 = \neg A_2$) の検出

制約矛盾 (チャイニーズウォールポリシー) の検出

次に，自由変数タブロー法を用いてチャイニーズウォールポリシーにより引き起こされる制約矛盾を検出可能であることについて述べる．図 3.13 に，チャイニーズウォールポリシーが原因の制約矛盾を検出した結果を示している．

一般的に，チャイニーズウォールポリシーにより引き起こされる矛盾は，次のように定義される．

定義 4 (制約矛盾 (チャイニーズウォールポリシー))**.** 以下に示す認可ポリシーの集合 Ω は，チャイニーズウォールポリシー cw と制約矛盾 (チャイニーズウォールポリシー) である．ここで，$1 \leq m \leq n, 1 \leq k_i \leq n, k_i \neq k_j (i \neq j)$ であり，C は任意の定数とする．

$$\begin{aligned} \mathrm{cw} &: \mathtt{CW}(all, \{T_1, T_2, \cdots, T_n\}_m, all) \\ \Omega &= \{\mathtt{Auth+}(S_C, T_{k_i}, A_C)\}_{i=1}^{m+1} \end{aligned}$$

また，認可ポリシー Auth+/-は，明示的に定義されている場合のみではなく，継承ポリシーにより暗黙的に導出される場合も含むものとする．

即ち，m 個以上の認可 (許可) ポリシーが定義される場合に，制約矛盾が発生することになる．制約矛盾 (チャイニーズウォールポリシー) に関しては，次の定理が成立する．

定理 4 (制約矛盾 (チャイニーズウォールポリシー) 検出の完全性)**.** 全ての制約矛盾 (チャイニーズウォールポリシー) は，自由変数タブロー法により検出することができる．

証明. 定義 4 の仮定から，m 個以上の認可 (許可) ポリシーが，異なる T_{k_i} に対して設定されていることになる．そこで，図 3.1 の (2) に示す写像 ζ を用いることにより，$P(S_C, T_{k_i}, A_C)$ という形式の論理式が，少なくとも m 個あることがわかる．次に，図 3.1 (13) を用いることにより，ポリシー cw から，以下の論理式を得る．

$$\bigvee_{i=1}^{n!/(m!(n-m)!)} (P_{i1}(x, T_1, y) \wedge \cdots \wedge P_{in}(x, T_n, y)) .$$

この論理式には，少なくとも m 個の否定論理式 $\neg P(x, T_{k_i}, y)$ が含まれる．そのため，自由変数タブロー法を用いて解析した場合に，すべての枝において必ず経路が閉じられることになる．従って，全ての制約矛盾 (チャイニーズウォールポリシー) が検出されるこ

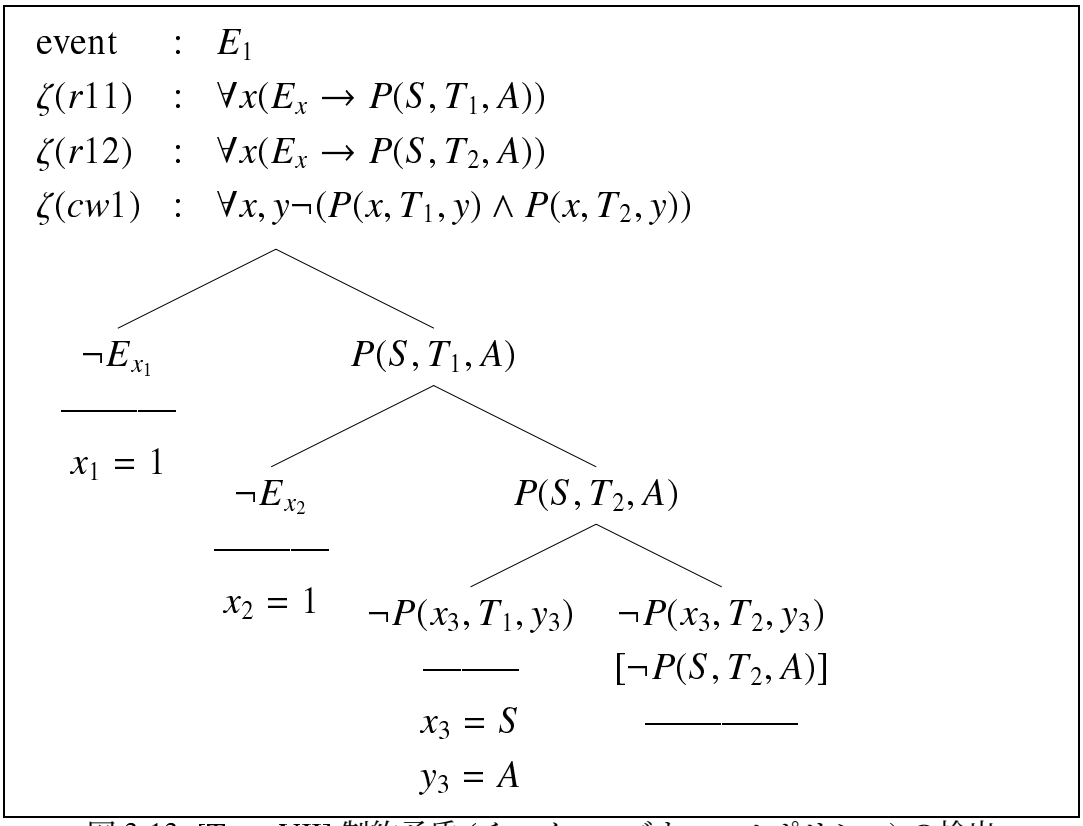

図 3.13　[Type VII] 制約矛盾 (チャイニーズウォールポリシー) の検出

とがわかる．□

職務分掌ポリシーに起因して発生する矛盾も，同様に自由変数タブロー法により検出することが可能であるが，チャイニーズウォールポリシーの場合と同様であるため，説明は省略する．

3.6　アクセス制御ポリシーの冗長検出

第 3.4 節で述べたように，自由変数タブロー法により，冗長なポリシーの存在も検出することが可能になる．例えば，次のアクセス制御ポリシーの組み合わせを考える．

$$\begin{aligned} \text{r28} &: \ \mathtt{prop}(\mathtt{Auth+}, \mathcal{H} \in \mathtt{SRH}, \mathtt{UP}) \\ \text{r29} &: \ \mathtt{Auth+}(S_1, T, A) \\ \text{r30} &: \ \mathtt{Auth+}(S_2, T, A) \end{aligned}$$

このときアクセス制御ポリシー r29 は，r28 及び r30 から導出されるため，冗長なポリシーである．このような冗長ポリシーの存在は，アクセス制御ポリシーの可読性を低下させるばかりではなく，システムの性能を低下させる可能性もあるため，アクセス制御ポリシーから削除されることが望ましい．

アクセス制御ポリシー r29 が冗長なポリシーであることを検出するためには，集合 $\{\zeta(\mathcal{H}), \zeta(r28), \zeta(r30), \neg\zeta(r29)\}$ に対して，自由変数タブロー法を用いることで検証できる．もしこの集合の解析の結果，全ての経路が閉じた場合には，$\zeta(r29)$ が，集合 $\{\zeta(\mathcal{H}), \zeta(r28), \zeta(r30)\}$ から帰結することを意味する．即ち，アクセス制御ポリシー r27 は冗長なポリシーであることがわかる．

一般的に，与えられたアクセス制御ポリシーの集合 $\{r_1, \cdots, r_n\}$ に対して，あるアクセス制御ポリシー r_i $(1 \leq i \leq n)$ が冗長であるか否かは，集合 $\{\zeta(r_1), \cdots, \neg\zeta(r_i), \cdots, \zeta(r_n)\}$ に対して，自由変数タブロー法を用いて解析することにより検証できる．

このように，自由変数タブロー法を応用することにより，矛盾するポリシーのみではなく，冗長なポリシーの検出も同様のアルゴリズムで行うことが可能となる．

3.7　アクセス制御ポリシーの整合性検証技術評価

本節では，これまで議論してきたアクセス制御ポリシー整合性検証技術を用いた場合に，実際のアクセス制御ポリシーの検証に要する時間がどの程度であるかを評価した結果について述べる．検証時間を評価するにあたり，タブロー法の実装として，leanCoP[39]を用いることとした．leanCoP は，最も基本的なタブロー法[29]の実装の一つであり，Prolog を用いて実装されている．leanCoP には矛盾の原因となる情報を推定する機能は実装されてはいないが，ポリシーの解析に要するおよその時間を評価するには十分な機能を備えている．

一般的に，解析時間は解析対象となるアクセス制御ポリシーの数や，アクセス制御ポリシーを論理式に変換したときの複雑性に大きく依存する．そのため，一律に解析時間を評価することを困難であるため，本節では次に示す 4 種類のモデルケースを用いて，評価を実施することとする．4 種類のモデルケースでは，それぞれ (i) 矛盾を含む集合と，(ii) 矛盾を含まない集合を準備した．

ケース I: (i) アクセス制御ポリシー r15 と r16 を含む単純矛盾 {Auth+ / Auth−}，あるいは，r19 と r20 を含む単純矛盾 {Obli+/Auth−} と，その他の無矛盾なポリシーで構成される集合.

(ii) 認可ポリシーと強制ポリシーのみを含む全てが無矛盾なアクセス制御ポリシーで構成される集合.

ケース II: (i) アクセス制御ポリシー r1, r2, pr1 と，主体ロール構造 $\mathcal{H}_s$ を含む暗黙的な単純矛盾 {Auth+ / Auth−} と，無矛盾なポリシーで構成される集合.

(ii) 認可ポリシーと継承ポリシーのみを含むすべてが無矛盾なアクセス制御ポリシーで構成される集合.

ケース III: (i) アクセス制御ポリシー ac1, r21, r22，r23 を含む制約矛盾（合成動作）及び，cw1, r11，r12 を含む制約矛盾（チャイニーズウォールポリシー）と，無矛盾なポリシーの集合.

(ii) 認可ポリシーと，合成動作ポリシーと，チャイニーズウォールポリシーを含む無矛盾なポリシーの集合.

ケース IV: (i) アクセス制御ポリシー prop1 と prop6 を含む暗黙的な単純矛盾と制約矛盾と，図 3.2 に示す主体ロールと客体ロールから構成される認可ポリシーと，チャイニーズウォールポリシー cw1 を含むアクセス制御ポリシーの集合.

(ii) 認可ポリシーと，継承ポリシーと，チャイニーズウォールポリシーを含む無矛盾なポリシーの集合.

表 3.1 評価環境のスペック

CPU	Pentium4 3.20GHz
Memory	1024MB
OS	Windows XP SP2
Prolog	SWI-Prolog Version 5.4.7

評価では，上で定義した 4 種類のモデルケースごとに，アクセス制御ポリシーの数が，2, 4, 8, 16, 32, 64, 128, 256, 512, 760, 1024, 1500, 2048 個となる集合を生成して，解析時間を計測した．また，それぞれのケースに関して，ポリシーを記述する順番をランダムに変更した 3 種類の集合を準備し，その平均解析時間を最終的な解析時間として算出した．ただし，いずれの場合においても，矛盾するポリシーの組み合わせが最も離れた位置に記述されるように配置を行っている．これはタブロー法による解析がその実装において上から順番に実施されることを考慮して，最長の場合の解析時間がどの程度になるかを評価できるようにするためである．

評価に使用したマシンスペックを表 3.1 に，またケース I からケース IV の評価結果を，それぞれ図 3.14〜図3.17 に示す．評価結果の図において，縦軸は矛盾検出に要した時間を，横軸は評価したポリシーの数を示している．また，例えば凡例 Auth+/Auth-(C) に示すような実線は，矛盾するポリシーを解析した場合の結果を，Auth+/Auth-(N) に示すような点線は，無矛盾な集合を解析した場合の結果をそれぞれ示している．

図 3.14 を見ると，単純矛盾の検出に関しては，アクセス制御ポリシーの数が 2048 個の場合であっても，25〜40 秒と比較的短時間で検証が完了していることがわかる．単純矛盾を検出するためには，単に定義されているアクセス制御ポリシーを比較するのみで検出可能であることが，評価時間を短くしている原因であると考えられる．一方で，図 3.15 に示すように，暗黙的な単純矛盾の検出にはより長い時間を必要としている．このことは，暗黙的な単純矛盾を検出するためには，タブロー法による解析の中で継承ポリシーを再帰的に分割していく必要があることが原因であると考えられる．即ち，ロールの構造や継承ポリシーがより複雑になると，解析時間もより長く必要となることが想定される．次に図 3.16 は合成動作やチャイニーズウォールポリシーにより引き起こされる制約矛盾の解析結果を示している．これらの矛盾の解析には再帰的処理が不要であるため，継承ポリシーを含む場合ほど，解析時間を必要としないことがわかる．

最後に図 3.17 は，継承ポリシーを含め，制約ポリシーなど様々な種類の矛盾が含まれ

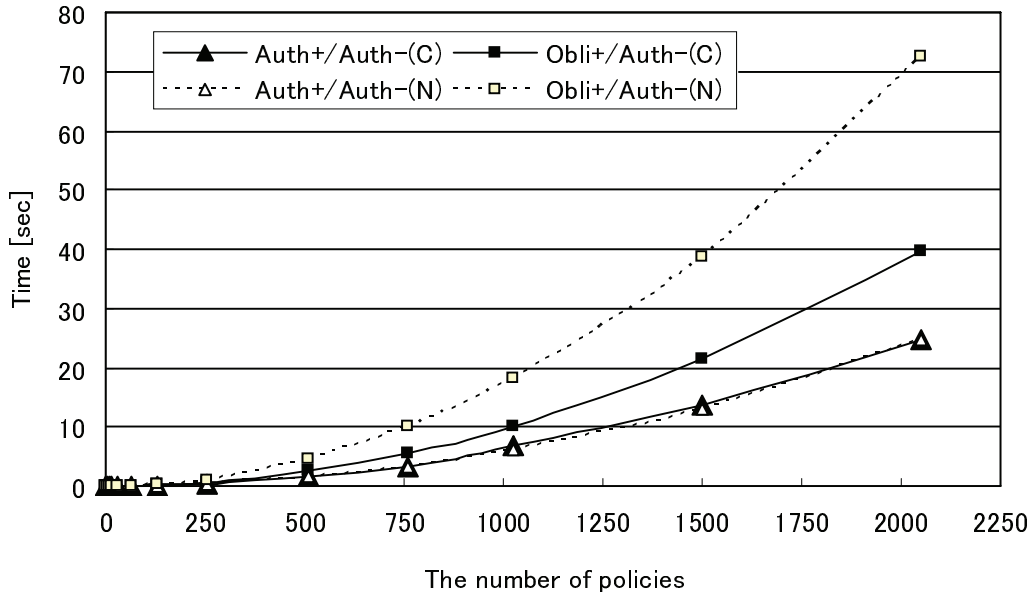

図 3.14 ケース I：単純矛盾の検出時間

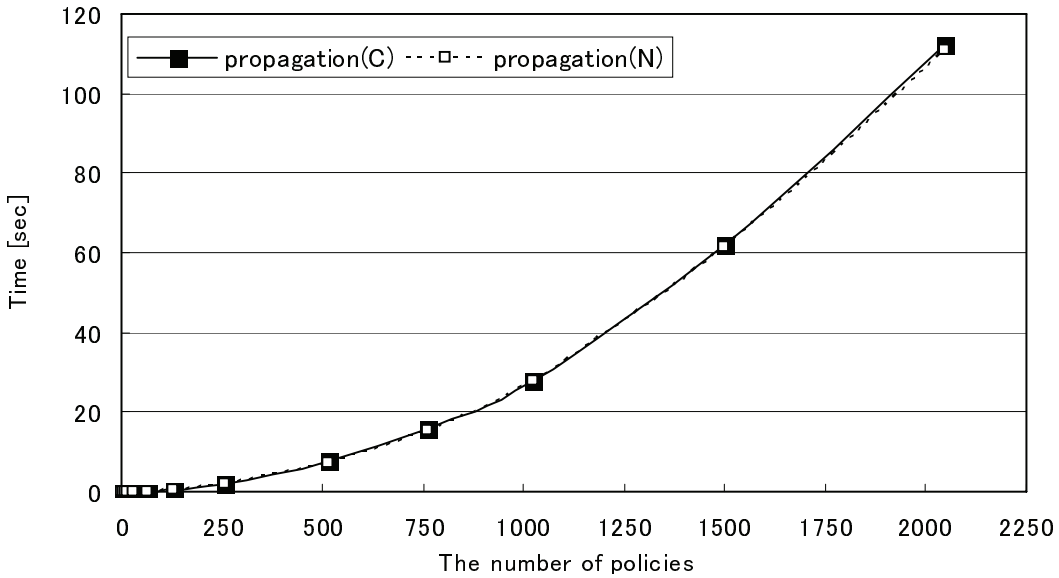

図 3.15 ケース II:暗黙的な単純矛盾の検出時間

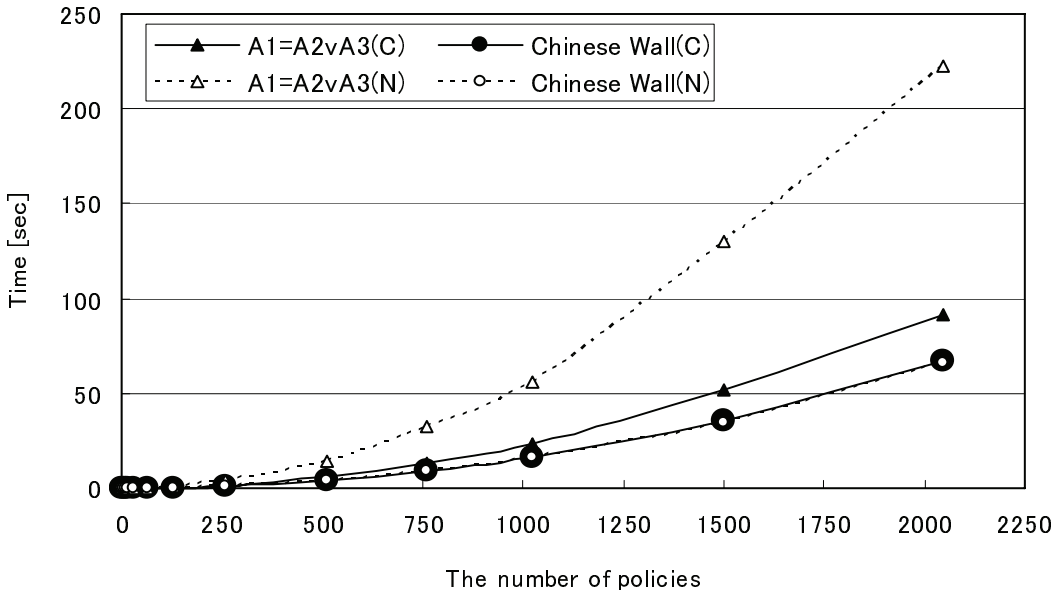

図 3.16 ケース III:制約矛盾の検出時間

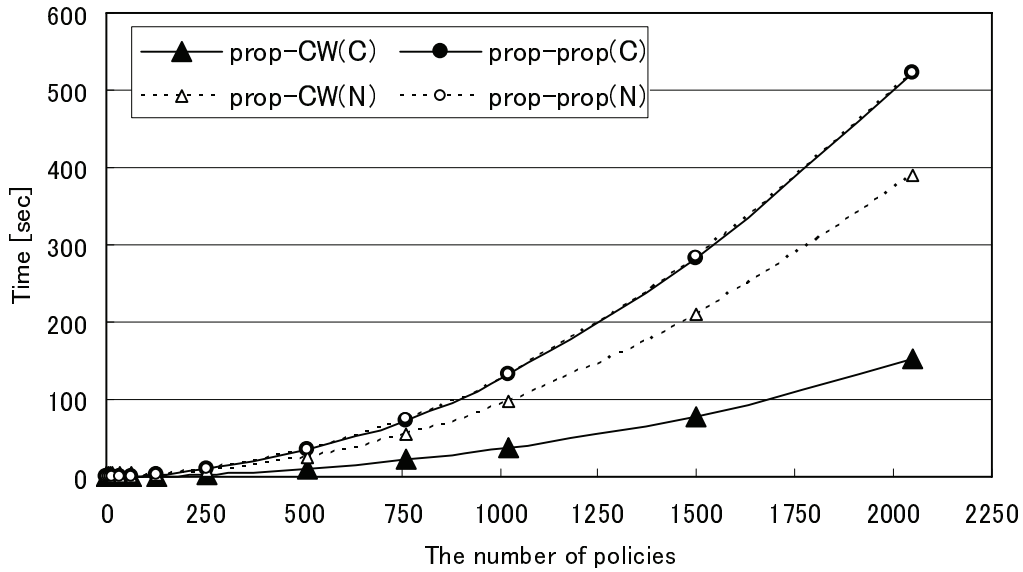

図 3.17 ケース IV：混合矛盾の検出時間

る場合の検証結果を示している．これまでの結果から容易に推定されるように，ケース 4 のポリシーの検証が，他のケースと比較して，最も時間を要していることがわかる．

これらの実験結果により，アクセス制御ポリシーの整合性検証技術により，アクセス制御ポリシーの整合性を検証可能であることが確かめられた．また，今回のモデルケースでは，1000 個程度のアクセス制御ポリシーの解析であれば，100 秒程度で解析が完了することも確かめることができた．アクセス制御ポリシーの整合性検証技術は，システム稼動前に静的にアクセス制御ポリシーを検証することを目的としているため，100 秒程度という解析時間は十分に実用的であるということができる．一方でポリシーの数が 1000 個を超える場合には，解析時間が大幅に長くなることは改善の必要があるといえる．しかしながら，定理証明のアルゴリズムを改良することにより，解析時間を短縮することは可能であると考えている．アルゴリズムの改良については，今後の課題である．

3.8 結言

本章では，まず最初に，アクセス制御ポリシーモデルに関して述べ，数学的な形式化を行った．次に，定義したアクセス制御ポリシーモデルにおいて発生するポリシーの矛盾と冗長性について定義した．そして，自由変数タブロー法を適用することにより，定義された矛盾と冗長性を検証するアクセス制御ポリシーの整合性検証技術について論じた．アクセス制御ポリシーの整合性検証技術を用いることにより，全ての種類の矛盾と冗長なポリシーを検出することが可能であることを証明した．最後に，シミュレーション評価を行い，アクセス制御ポリシーの検証に要する時間について検証した．評価の結果，アクセス制御ポリシーの整合性検証技術がある程度大規模なシステムにおいても，実用性が十分にあることを確認できた．

第 4 章

アクセス制御ポリシーの生成技術

4.1 緒言

第 3 章では，システムに設定されたアクセス制御ポリシーに含まれる矛盾するポリシーや冗長なポリシーを自由変数タブロー法を用いて検出することにより，アクセス制御ポリシーの安全性を確保する技術について論じた．しかしながら，アクセス制御ポリシーを正しく記述することができても，その内容がシステムに正しく反映されなければ，セキュリティを確保することはできない．本章では，アクセス制御ポリシーをシステムに正しく安全に反映する技術について論じる．

アクセス制御ポリシーには，第 3 章で述べたように認可ポリシーや強制ポリシーなど，いくつかの種類が存在する．例えば，認可ポリシーは，次のように記述されると述べた．

Auth+(医師, カルテ, 編集)

これは，「“医師”は，“カルテ”を，“編集”することができる」という意味である．この認可ポリシー自身の意味は，簡単な内容であり，直感的に理解することができる．しかしながら，このアクセス制御をシステム上で実現するためには，ユーザを管理するシステム，カルテを管理するシステム，ユーザとカルテを管理するシステムとを接続するネットワークなど，様々なシステムを整合的かつ安全に制御することが必要であり，それを実装することは容易ではない．また医療分野に関しては，近年の医師不足などにより，一人の医師が複数の病院で診療活動を行う場合があり，アクセス制御の対象は同一病院内のシステムのみに留まらず，異なる病院間のシステムを対象とする必要があることも問題を

難しくしている．さらに，医療現場では，患者の生命に関わる情報を扱うため，アクセス制御の安全性に加え，一刻も早く情報にアクセスできる迅速性も求められる．

そこで，本章では，医療分野を対象として，異なる病院間の医療機器や医療情報に安全かつ迅速にアクセスするために必要となる技術課題について整理し，それを解決するための，アクセス制御ポリシーの生成技術について論じる．

4.2　アクセス制御ポリシーの自動生成に関する技術課題

インターネットを利用可能な携帯電話や PDA などの小型端末の普及により，いつでもどこからでもネットワークを介したサービスを利用可能なユビキタス環境が整い，病院などでも利用が進みつつある．また，従来はセキュアな通信路を確保するために高価な専用線を利用する必要があったが，IPsec[60]などの暗号化技術を用いた VPN (Virtual Private Network) の普及により，セキュアな通信路を安価に実現することが可能となった．そのため，これらの技術を用いることにより，セキュアなユビキタス環境が実現可能となりつつある．

しかしながら既存の技術を用いて，医療情報という極めて高い機密性を求められる情報を扱う病院間でセキュアなユビキタス環境を構築するためには，解決しなければならない技術的課題が 2 つ存在する．

第一に，VPN 接続を行うために必要な情報を事前に特定することができない点が課題の一つとしてあげられる．一般に VPN 接続を行うためには，VPN 通信を行う接続元・接続先双方のネットワーク機器に対して，暗号化アルゴリズムや鍵情報などの複雑かつ機器や使用されるシステムに固有な情報を，事前に設定しておく必要がある．これまでは，例えばある病院のネットワークとその病院に勤める医師の自宅の端末を接続するといった具合に，接続元と接続先機器の IP アドレスや使用する暗号化アルゴリズムなどの VPN 接続に必要な情報はある程度既知であることが前提であった．そのため，VPN 接続の設定作業は煩雑ではあるものの，大きな障害とはならなかった．一方で，外出先や異なる病院内からのアクセスが行われるユビキタス環境では，接続元と接続先機器の IP アドレスや暗号化アルゴリズムなどの VPN 接続に必要となる具体的な情報が事前に特定されているとは限らない．従って，事前に VPN 接続に必要な情報を通信機器に設定しておくことは不可能である．そこで，任意の 2 地点間で VPN 接続に必要な構成情報を要求に応じて，アクセス制御ポリシーから自動的に生成するための技術が必要となる．

第二に，端末の小型化に伴い端末の盗難によるなりすましがユビキタス環境では脅威となる．従来よりなりすましの解決策のひとつとして公開鍵証明書による認証方式が利用されているが，公開鍵証明書の保存された端末自体が盗難にあった場合には，なりすまされる可能性が高くなるという課題がある．いくらアクセス制御ポリシーが適切に設定され，必要に応じて自動的に機器固有の構成情報に変換されることが実現されたとして

も，その端末自身が不正に利用されるようなことがあれば，安全性を確保することはできない．医療分野においてセキュアなユビキタス環境を構築するためにはこれらの課題を解決する必要がある．

第一の課題を解決する方式として例えば，A. Bandara ら[1] は，QoS 分野において，KAOS[28] アプローチを応用することにより各ネットワーク機器の構成情報を自動的に導出する手法に関して提案している．しかしながら，VPN の構成情報生成に関しては言及されていない．また，Z. Fu ら[16] は接続元と接続先がネゴシエーションすることにより，IPsec-VPN 構成のための暗号化や認証方式の情報を自動的に生成する方式について提案している．H.B. Wang ら[47] は，設定された IPsec のセキュリティポリシーの矛盾を検出し，自動的に修正する方式について提案している．しかしながら，どちらの方式も接続元と接続先は既知であるか信頼されている相手であることが前提である．ユビキタス環境では，接続先が未知であっても，相手が信頼できる相手であるかどうかを確認した上で接続できることが重要であり，これらの方式では対応することが難しい．

また，第二の課題を解決する方式として，馬場ら[61] は，組織内のネットワークに不正な端末が接続されたことを検知し，自動的に隔離するための方式を提案している．しかしながら，既に認証されたネットワーク内の端末が不正に持ち出され，外部で悪用されることを防ぐことまではできない．

そこで，本章では，オンデマンド VPN 技術[53] を用いて，これら二つの課題を解決する方式について論じる．オンデマンド VPN 技術は，第一の課題を VPN 接続に必要となる IPsec 構成情報をアクセスセス制御ポリシーから自動的に生成技術により，そして，第二の課題を 2 階層 PKI (Public Key Infrastructure:公開鍵基盤) 技術に対応した耐タンパ IC チップによる厳密な機器認証技術により解決する．耐タンパ IC チップとは，IC チップの内部の動作や処理手順が外部に漏れることがないように，機器や回路の中身を外部から分析しにくくすることで，内部解析（リバース・エンジニアリング）や改変に対する防護性を高めた IC チップのことである．

第 4 章の次節以降の構成は次の通りである．第 4.3 節では，オンデマンド VPN システムを構成する主要技術である，2 階層 PKI 技術とアクセス制御ポリシーの生成技術について述べる．第 4.4 節では，これら二つの技術を用いてどのようにオンデマンド VPN システムのサービスが提供されるかについて述べる．そして，第 4.5 節では，オンデマンド VPN のプロトタイプシステムを用いた性能評価結果と運用評価結果及び今後の課題について述べ，最後に第 4.6 節で本章のまとめを述べる．

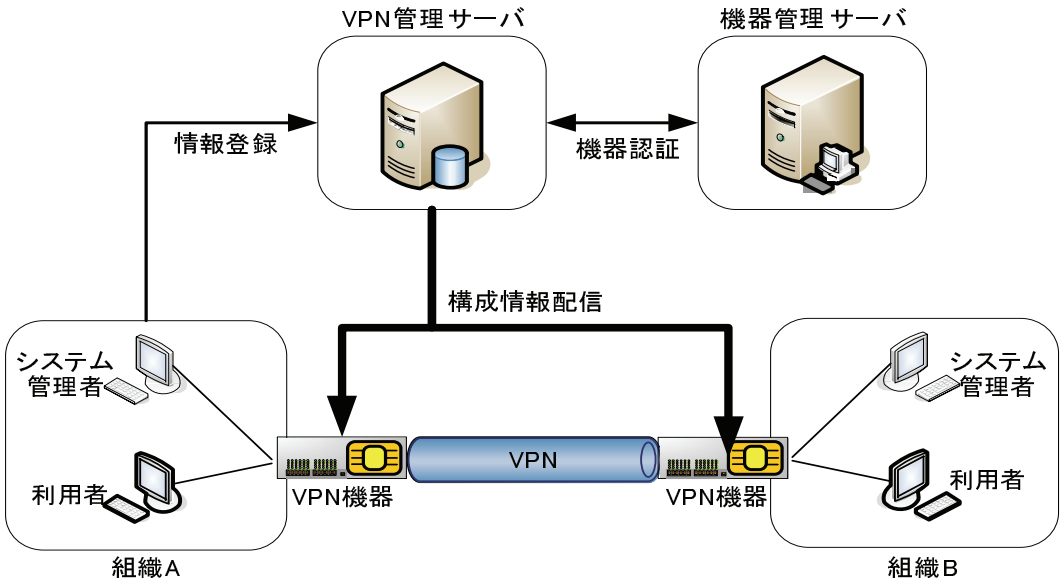

図 4.1 オンデマンド VPN システム全体構成

4.3　2 階層 PKI 技術とアクセス制御ポリシーの生成技術

4.3.1　システム構成

オンデマンド VPN システムは，機器管理サーバ・VPN 管理サーバ・VPN 機器から構成されるシステムである．オンデマンド VPN システムの全体構成を図 4.1 に示す．機器管理サーバは VPN 機器の真正性を保証するためのサーバである．VPN 管理サーバは，機器管理サーバに認証された機器による VPN 接続を制御するサーバである．VPN 機器は実際に VPN コネクションを開設する機器であり，一般にはルータが VPN 機器となる．機器管理サーバ，VPN 管理サーバ，VPN 機器の機能構成図をそれぞれ図 4.2，図 4.3，図 4.4 に示す．機器管理サーバ・VPN 管理サーバ・VPN 機器間の通信には SSL を用い，機器間の相互認証を行うと共に，送受信するメッセージを盗聴・改ざんから保護する．

オンデマンド VPN システムは，これらのサーバ・機器が協調しセキュアなユビキタス環境を実現するシステムである．オンデマンド VPN システムで実装される主要な技術として 2 階層 PKI 技術，アクセス制御ポリシーから IPsec 構成情報生成する，アクセスセス制御ポリシー生成技術がある．次に，これらの技術と実装方式について述べる．

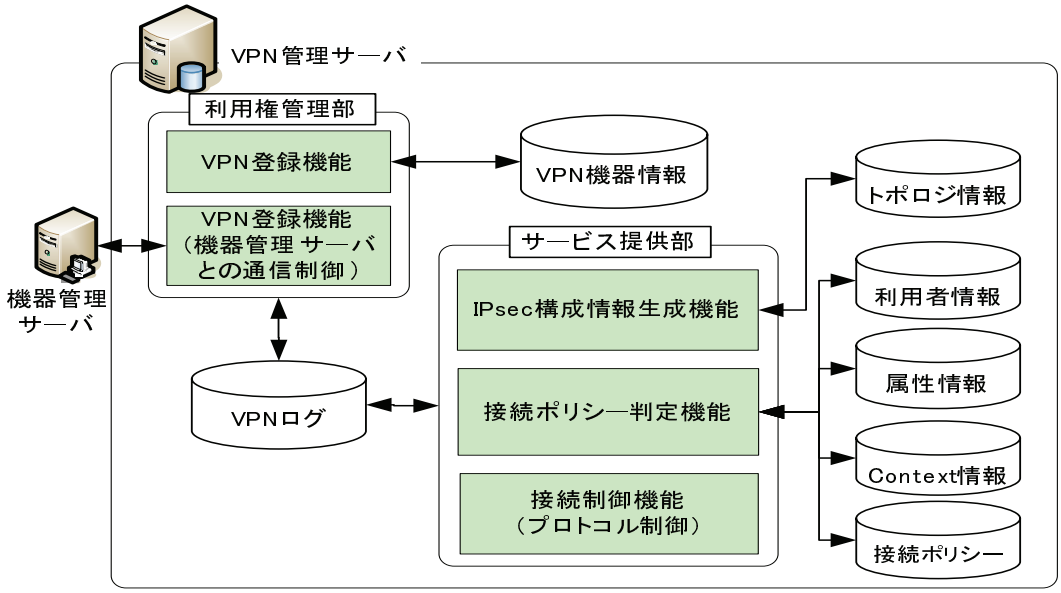

図 4.2 VPN 管理サーバの機能構成

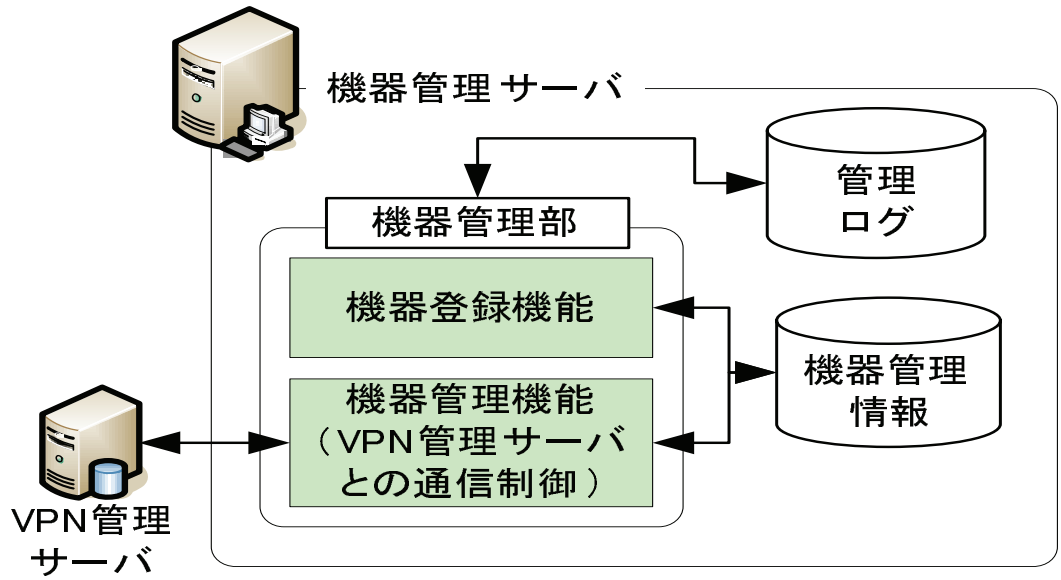

図 4.3 機器管理サーバの機能構成

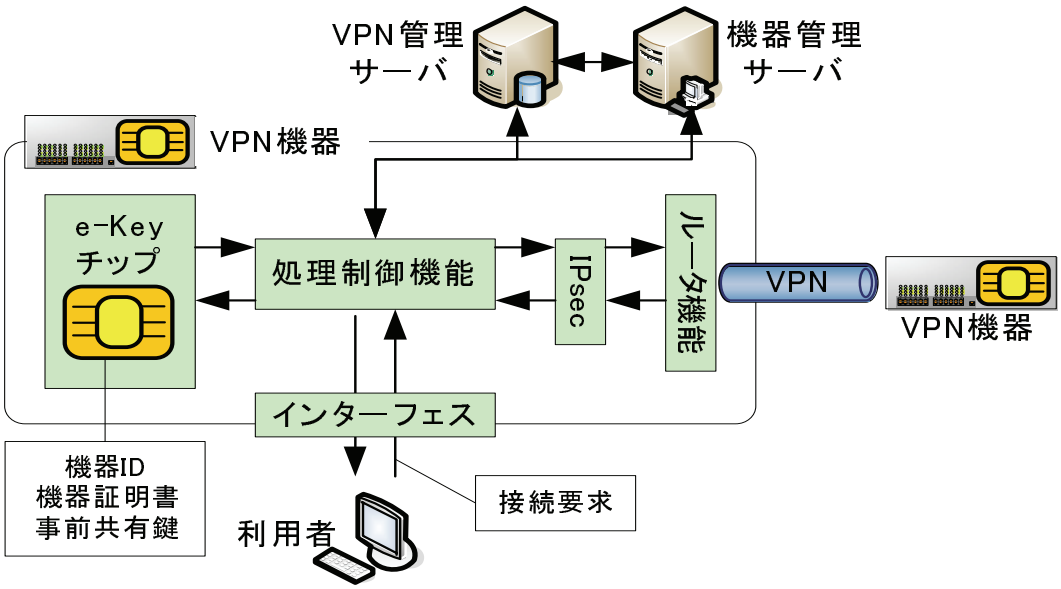

図 4.4 VPN 機器の機能構成

4.3.2　2 階層 PKI 技術

オンデマンド VPN システムでは IPsec を用いた VPN を構築する．IPsec ではパケットの暗号化を行うための鍵交換に IKE (Internet Key Exchange)[18]を利用することが一般的である．IKE は相手認証，SA (Security Association) の折衝と管理，共有秘密鍵管理などを行うプロトコルであり，暗号化を行うための鍵交換を行うこともできる．従来，IKE を行うために必要となる情報を，VPN 接続を行う相互のルータに事前に設定しておく必要があった．そのため，これらの情報が漏洩すると機器のなりすましが可能となり，機密情報の漏洩につながるという危険性があった．そこでオンデマンド VPN システムでは，IPsec による VPN 接続に必要な情報を VPN 管理サーバで一元的に管理し，利用者の要求に応じて VPN 機器に配信するセンタ型システムとすることで事前にルータに情報を設定しておくことを不要とした．また，耐タンパデバイスである e-Key チップを VPN 機器に組み込むことにより，機器のなりすましを防止すると共に，初めてデータをやり取りする相手とも安全な通信を行うことを可能にした．本項では e-Key の仕組みについて述べる．

e-Key チップは Secure e-Key Network (SeKNW) のフレームワークの中で情報流通機器に内蔵された状態で販売されることを前提とされている．SeKNW は 2 階層 PKI 技術を

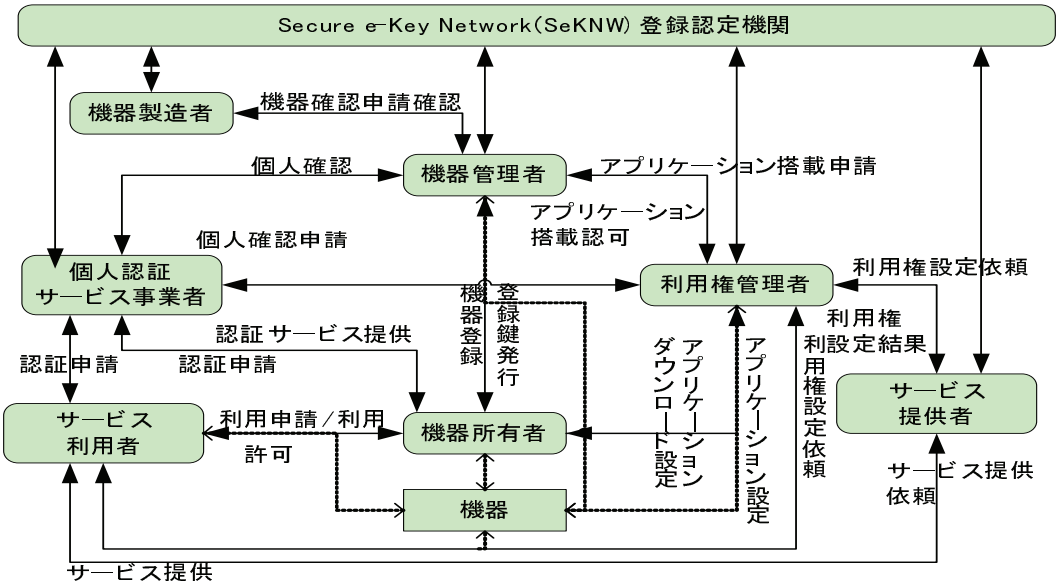

図 4.5 Secure e-Key Network (SeKNW) フレームワーク

ベースとした IC チップの管理運用モデルの一つであり，NICSS フレームワーク[54]を参考に策定されたものである．SeKNW のフレームワークを図 4.5 に示す．2 階層 PKI 技術は，1 階層目の PKI を利用して e-Key チップ発行後でも自由にアプリケーションを発行・設定し，各チップアプリケーションが独自に 2 階層目の PKI を利用してサービスを提供できるなど，サービス間のセキュリティを保ちながら幅広いサービスが提供可能な技術である．オンデマンド VPN では，1 階層目の PKI 情報を VPN 機器の認証のために，2 階層目の PKI 情報を VPN サービスの利用権認証のためにそれぞれ利用することにより，安全性の高いサービスを容易に実現することを可能としている．

4.3.3　アクセス制御ポリシーの生成技術

オンデマンド VPN システムは VPN 管理サーバ上で VPN 接続に必要となる IPsec 構成情報をアクセス制御ポリシーと VPN 接続要求に含まれる情報から動的に生成・配信することにより，VPN コネクションを構築する．また，オンデマンド VPN では IPsec の暗号化処理に Linux 上で動作可能な FreeS/WAN[30]を利用している．VPN 管理サーバ上で構成し VPN 機器に配信しなければならない IPsec 構成情報は大きく 4 つある．以下にそれぞれの情報について述べる．

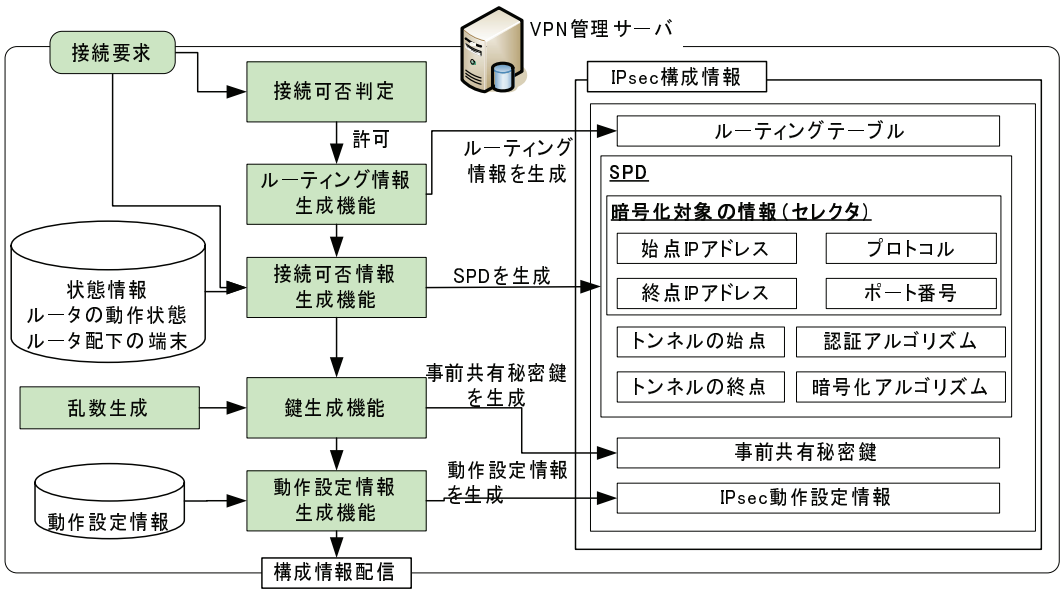

図 4.6 アクセス制御ポリシーの自動生成処理

(1) **セキュリティポリシーデータベース (SPD)：**SPD は，IPsec による暗号化処理対象パケットを選別するための情報が記述されたデータベースであり，VPN 機器から VPN 管理サーバへ通知された IP アドレスを元に生成・配信する．具体的には，VPN 機器の配下に存在する VPN 接続要求を行った利用者端末の IP アドレスと，VPN 接続要求先端末の IP アドレスが記述された SPD を生成する．VPN 機器は，パケットを受信する毎に SPD を検索し，そのパケットが暗号化の対象となるパケットであるか否かの判定を行う．オンデマンド VPN では，SPD により明示的に暗号化が指定されたパケットのみを暗号化処理の対象とする．配信された情報は ipsec.conf の中に記述される．

(2) **IPsec 動作設定情報：**IPsec 動作設定情報とは，機器間の IPsec コネクションである SA を生成するために必要となる情報である．例えば，認証方式や暗号化アルゴリズム，SA の有効期間などの情報が含まれる．FreeS/WAN の場合 IKE で使用するアルゴリズムは変更できなため，ipsec.conf に記述するべき情報をサーバ内に静的に設定しておき，接続要求があった場合に，IPsec 動作設定情報を生成・配信する．

(3) **事前秘密共有鍵：**オンデマンド VPN では IKE を行う際の認証方式として事前共有秘密鍵方式を用いる．そのために必要となる鍵は，VPN 管理サーバで接続要求毎に乱数を用いて新規に作成され，SSL 経由で VPN 機器の e-Key チップへと書き

込まれる．管理サーバではこの事前共有秘密鍵のみを生成し，実際の暗号通信路に使用される秘密鍵は VPN 機器同士が VPN 接続を行っている時にしか知り得ないようにしている．

(4) **ルーティングテーブル：**ルーティングテーブルは IP パケットの経路情報を記述したものである．VPN 接続の相手先への IP パケットが，適切な暗号通信路を経由するように VPN 管理サーバがルーティングテーブルを生成し，各 VPN 機器に配信する．配信された情報は iptables.conf に反映される．

VPN 管理サーバの IPsec 構成情報生成に関する機能構成図を図 4.6 に示す．アクセス制御ポリシー生成技術により，事前に VPN 接続に必要となる情報を VPN 機器に設定することなく，オンデマンドに VPN 接続を実現することが可能となる．

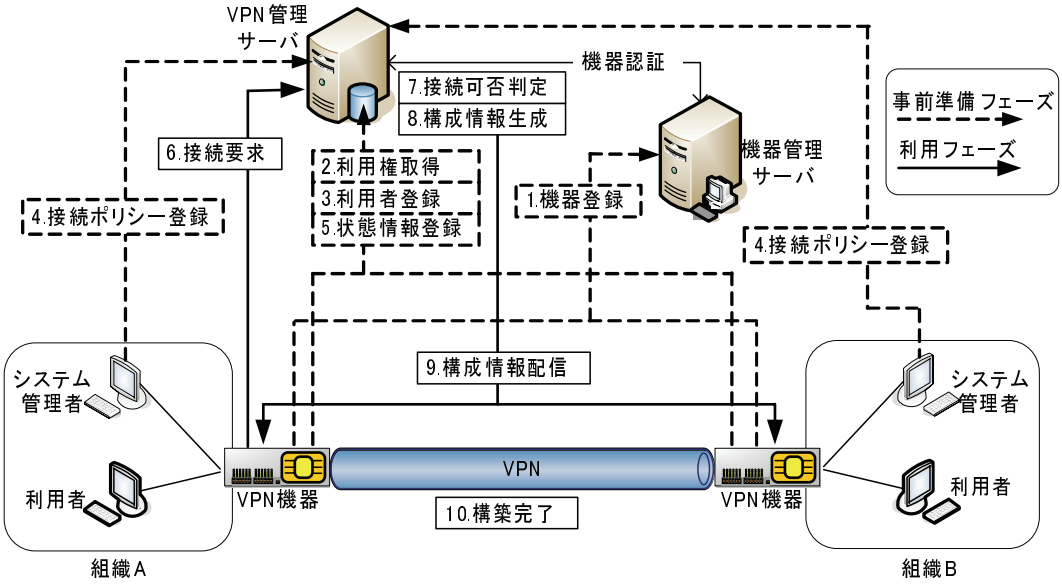

図 4.7 オンデマンド VPN システム構成図

4.4　サービス提供手順

本節では，前節で述べた 2 階層 PKI 技術と，アクセスセス制御ポリシー生成技術により，オンデマンド VPN システムがどのようにサービスを提供するかについて述べる．オンデマンド VPN システムを利用するためには，大きく事前準備フェーズと利用フェーズの二つのフェーズが必要となる．事前準備フェーズはオンデマンド VPN サービスを利用するために必要な環境を構築するフェーズであり，利用フェーズは構築された環境を利用してオンデマンド VPN サービスを利用するフェーズである．図 4.7 にその概要を示す．図 4.7 を用いてオンデマンド VPN システムのサービス提供手順について述べる．

4.4.1　事前準備フェーズ

オンデマンド VPN システムを利用するためには，事前に SeKNW 登録認定機関により認可された製造者が製造した VPN 機器を利用する必要がある．VPN 機器には e-Key チップが搭載されており，機器の製造者が VPN 機器製造時に e-Key チップ内に正当な機器であることを証明するための仮の公開鍵証明書を保存しておく．次にオンデマンド

VPN システムを利用する利用者が所属するネットワークのシステム管理者がこの VPN 機器を購入し，オンデマンド VPN システムを利用するための登録・設定を行う必要がある．VPN 機器の登録までの処理が事前準備フェーズとなる．事前準備フェーズで登場する各プレイヤの役割を表 4.1 に示し，事前準備フェーズの詳細を以下に述べる．

(1) **機器登録：**SeKNW 登録認定機関より認定された VPN 機器管理者の管理するサーバを機器管理サーバとする．最初に VPN 機器所有者は VPN 機器を用いて機器管理サーバにアクセスし，VPN 機器の登録を行う．VPN 機器管理サーバは，事前に VPN 機器の e-Key チップ内に保存された仮の公開鍵証明書を用いて VPN 機器の認証を行う．認証後 VPN 機器と機器管理サーバ間で正式な公開鍵証明書・秘密鍵を作成し，チップ内の仮の証明書と置き換える．これらを 1 階層目の PKI 情報と呼ぶ．その後機器管理サーバは，VPN 機器から送信された VPN 機器の設置場所や所属組織などの機器情報及びシステム管理者の情報登録を行う．機器情報を登録することにより機器の盗難によるなりすましを防止することが可能となる．

(2) **利用権取得：**次に，システム管理者は VPN サービス提供者に VPN サービスの利用申請を行う．VPN 利用権管理者は VPN 機器管理者にチップアプリケーション

表 4.1 オンデマンド VPN のプレイヤ

VPN 機器 管理者	VPN 機器に搭載された e-Key チップ上の資源を管理する
VPN 機器 所有者	e-Key チップを搭載した VPN 機器を所有・管理する
VPN サービス 提供者	オンデマンド VPN サービスを 提供する
サービス 利用者	オンデマンド VPN サービスを 利用する
システム 管理者	オンデマンド VPN サービスの 利用申請・利用者登録などを行う
VPN 利用権 管理者	VPN サービスを利用するための アプリケーションと利用権 の発行・管理を行う

の搭載許可を得る．そして，VPN 機器管理者によって構築されるセキュアチャネルを用いて VPN サービス利用のためのチップアプリケーションを e-Key チップ内にダウンロードし 2 階層目の PKI 情報を保存する．

(3) **利用者登録：**システム管理者は実際に VPN サービスを利用する VPN 利用者情報を VPN 管理サーバへ登録する．VPN 管理サーバは 2 階層目の PKI 情報を用いて VPN 機器を認証し，送信されてきた利用者の ID/パスワードなどの情報を登録する．

(4) **接続ポリシー登録：**システム管理者は，VPN 利用者が VPN 接続を行うことを許可・禁止する相手先情報，及び自ネットワークへの接続を許可・禁止する相手先の情報をそれぞれ受信用アクセス制御ポリシー・発信用アクセス制御ポリシーとして登録する．

(5) **状態情報登録：**VPN 機器は，VPN 機器自身と VPN 機器配下に存在する VPN サービスを利用する端末の IP アドレスや端末種別などの状態情報を取得する．VPN 機器は VPN 管理サーバへ状態情報を通知し，VPN サーバはこの情報をデータベースへ登録する．登録された状態情報は公開され VPN 接続先を選択する際に利用される．

4.4.2 利用フェーズ

事前準備フェーズで登録された情報を元に，VPN 接続を行うのが利用フェーズである．VPN サービスを利用したい VPN 利用者が VPN 管理サーバに VPN 接続の要求を行ってから，VPN が構築されるまでのフェーズが利用フェーズの対象である．下記の (6)〜(10) に利用フェーズの詳細を述べる．

(6) **接続要求：**オンデマンド VPN 利用者は接続したい VPN 機器を選択し接続要求を行う．オンデマンド VPN 機器は利用者の要求に対して VPN 管理サーバへ接続を行い 2 階層目の PKI 情報による利用権認証を実施した上で，接続先検索の機能を提供する．接続先の情報は事前準備フェーズで登録された状態情報から取得する．

(7) **接続可否判定：**VPN 管理サーバは接続要求と受信用アクセス制御ポリシー及び発信用アクセスセス制御ポリシーをそれぞれ照合し，要求された VPN 接続が許可される要求か禁止される要求かを判定する．接続が禁止される場合には，VPN の構

築作業を中止する．

(8) **構成情報生成：**接続が許可される場合には，状態情報と利用者の接続要求及びアクセス制御ポリシーから VPN 構成に必要となる IPsec 構成情報と，IKE 用の事前共有秘密鍵を生成する．

(9) **構成情報配信：**VPN 管理サーバから構成情報と事前共有秘密鍵を接続要求元・要求先それぞれの VPN 機器へ配信し，搭載されている e-Key チップへ格納する．

(10) **構築完了：**受信した構成情報と事前共有秘密鍵を用いて IKE を実行し，VPN を構築する．以降，接続元と接続先は開設された VPN を経由した通信が可能となる．VPN 通信完了時には切断要求をあげ VPN コネクションを破棄する．

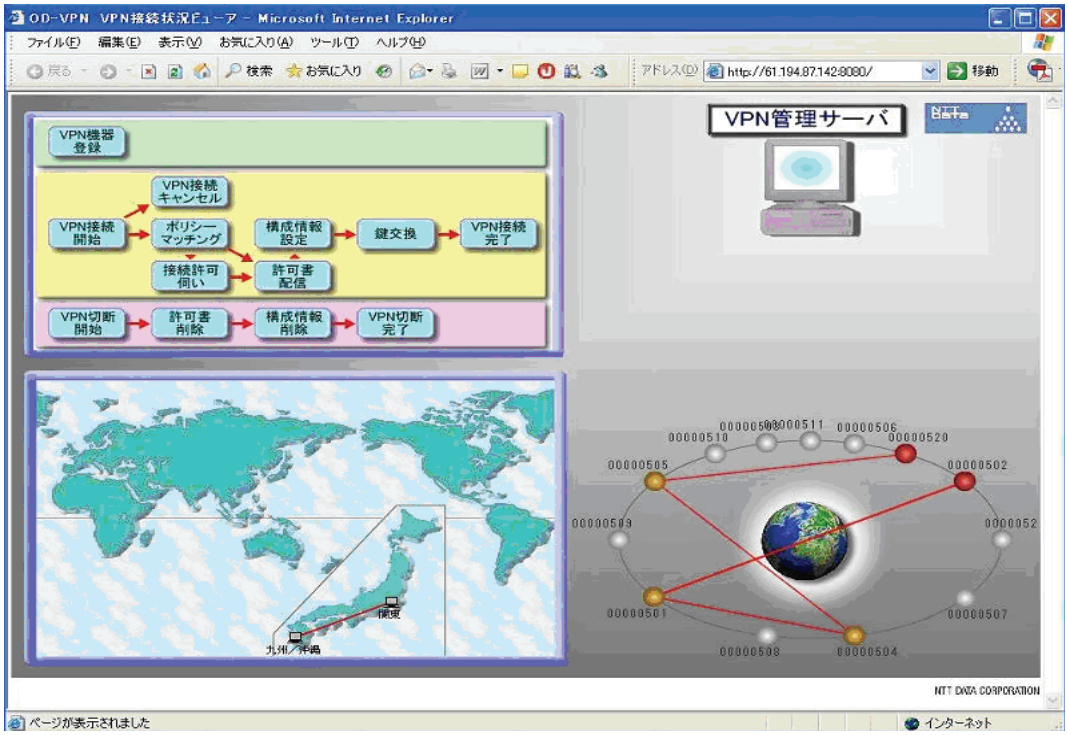

図 4.8 VPN 管理サーバ画面

4.5　アクセス制御ポリシーの生成技術評価

第 4.3 節～第 4.4 節で述べた各機能を実装した機器管理サーバ・VPN 管理サーバ・VPN 機器のプロトタイプをそれぞれ作成した．機器管理サーバと VPN 管理サーバの各機能は Red Hat Enterprise Linux ES 上で実装を行った．また，Linux カーネルを搭載し e-Key チップを内蔵したルータを開発し，VPN 機器として利用した．VPN 管理サーバには管理者が VPN の接続状況を監視することができるための VPN 接続ビューワ（図 4.8 参照）を実装した．さらに，VPN 機器には VPN 利用者が接続要求を行うことができるようにするための Web アプリケーションを実装した．接続要求アプリケーションは，VPN 管理サーバと連携し VPN 接続先として登録されている機器を自動的に取得することにより，VPN 利用者がブラウザから接続先を選択し容易に VPN 接続要求を行うことを可能にする．これら一連のプロトタイプを用い，利用フェーズにおける性能評価として VPN 接続時間の評価・VPN による転送効率の評価・アクセス制御ポリシー可否判定時間の評価を

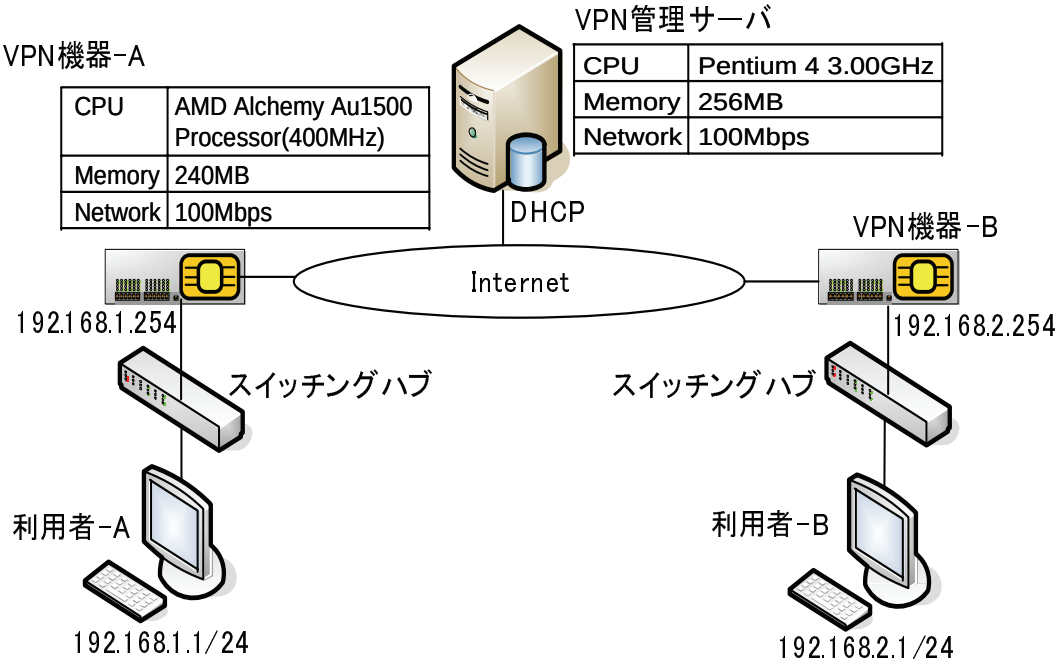

図 4.9 VPN 接続時間測定実験環境

それぞれ実施した．さらに，フィールド評価を実施し事前準備フェーズおよび利用フェーズの利便性評価を実施した．本節ではこれらの評価内容と結果・考察について述べる．

4.5.1　VPN 接続時間の評価

本項では，VPN 接続時間の評価方法と結果・考察について述べる．評価を行った実験環境を図 4.9 に示す．図 4.9 において VPN 機器-A と VPN 機器-B は同じ性能である．開発したプロトタイプにより VPN が接続されるまでに必要とする時間を評価するために，以下の項目 (1)〜(4) に関して VPN 接続に必要な手順毎に処理に要する時間をそれぞれ測定した．VPN 接続の手順と各項目の関係を図 4.10 に示す．

(1) **ユーザ側応答時間：**利用者-A が VPN 機器-A に接続し，利用者-B の端末への VPN 接続開始ボタンを押下してから VPN 接続確認ダイアログが表示されるまでの時間．

(2) **ルータ側応答時間：**VPN 管理サーバ上で VPN 接続要求を受信してから VPN 接続完了通知を返信するまでの時間．

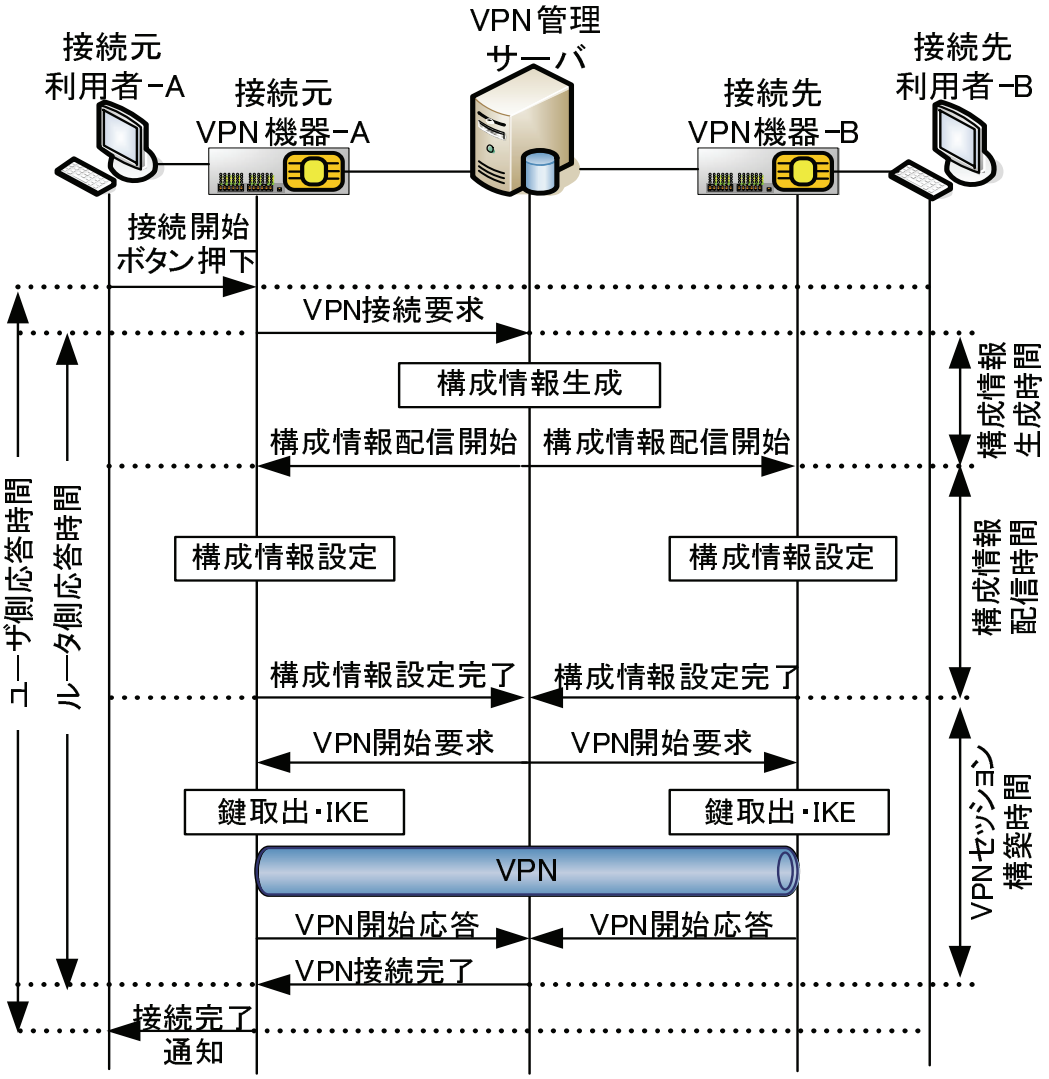

図 4.10 VPN 接続手順

(3) **VPN 接続元への構成情報配信時間：**VPN 管理サーバが接続元の VPN 機器に対して，IPsec 構成情報配信を開始してから，VPN 機器で構成情報が設定され，構成情報設定完了通知を受信するまでの時間.

(4) **VPN 接続先への構成情報配信時間：**VPN 管理サーバが接続先の VPN 機器に対して，IPsec 構成情報配信を開始してから，VPN 機器で構成情報が設定され，構成情報設定完了通知を受信するまでの時間.

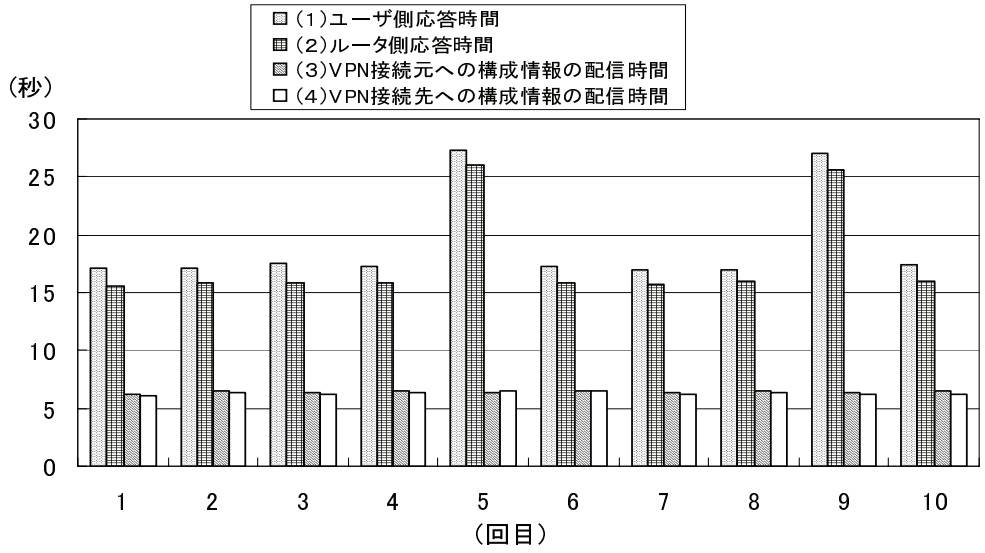

図 4.11 VPN 接続時間測定結果

VPN 接続要求を開始してから VPN 接続が完了するまでの処理を 10 回実施し，それぞれの処理に要した時間を計測した．実験結果を図 4.11 に示す．ルータ側応答時間は，10 回中 8 回は約 15.8 秒，2 回は約 25.9 秒と測定結果にばらつきが生じた．応答時間のばらつきの原因を調査した結果，今回のプロトタイプに使用した FreeS/WAN の IKE の実装方法に原因があることが明らかとなった．FreeS/WAN の実装では，接続処理を頻繁に行うと IKE のネゴシエーション時にリトライ処理が発生する．このことが計測時間のばらつきの原因となっていた．リトライの間隔と回数は，FreeS/WAN の設定ファイルで変更可能であり調節することにより処理に要する平均時間を短縮することが可能であると考えられる．

リトライ処理が発生しなかった場合の各処理毎の平均処理時間と，ユーザ側応答時間を 1 とした場合の平均処理時間比率を図 4.12 に示す．実験の結果，リトライ処理がない場合の VPN 接続処理に要するユーザ側応答時間は平均で約 18 秒であった．また，VPN 管理サーバで VPN 接続要求を受信してから構成情報の配信を開始するまでの構成情報生成時間，VPN 接続元への構成情報配信時間と VPN 接続先への構成情報配信時間の平均である構成情報配信時間，VPN 機器がチップから鍵情報を取り出し IKE のネゴシエーションにより VPN 接続が完了するまでの VPN セッション構築時間は，それぞれ平均で 4.1 秒，6.4 秒，5.3 秒であった．

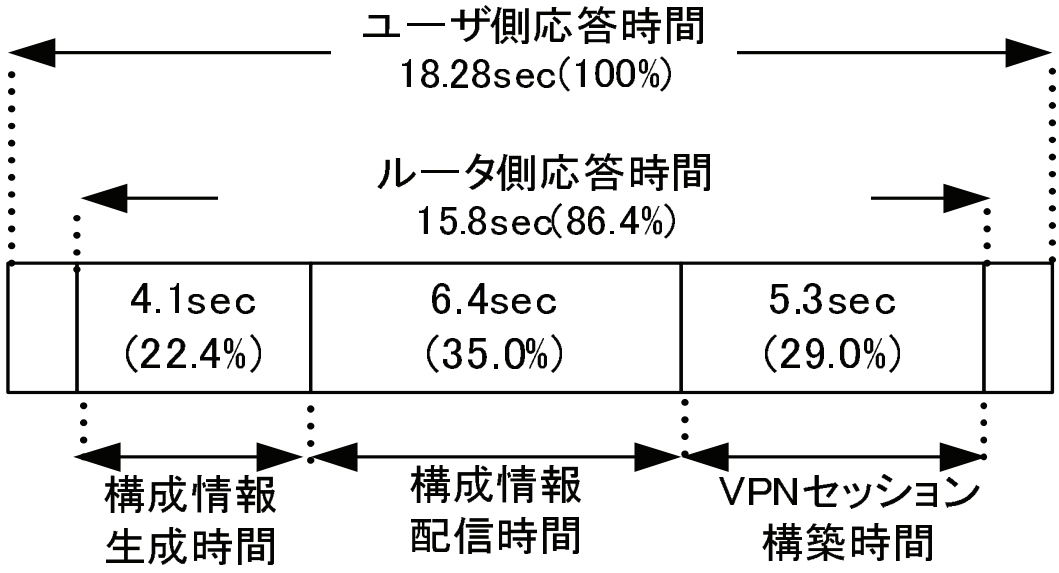

図 4.12 各処理毎の VPN 接続時間比率

オンデマンド VPN の適用先としては，前述したように医療現場での遠隔診療などを想定している．例えば遠隔診療を行う場合の VPN 接続利用時間は数十分から数時間である場合が一般的である．今回の実験により得られた VPN 接続時間は，VPN 接続利用時間と比較して相対的に十分小さい．従って遠隔診療を行う場合には，VPN 接続時間の実運用上の影響は少ないと考えられる．しかしながら，例えば外出先からメールチェックのみを行うために VPN を利用する場合には，その利用時間は数十秒から数分である．従って VPN 接続に 18 秒を要するシステムが利便性に与える影響を無視することはできず，適用分野によっては接続時間の改善が必要である．

4.5.2　VPN による転送効率の評価

医療分野では遠隔病理診断のために顕微鏡画像の送受信が行われる．正確な診断のためには，マルチスペクトル顕微鏡を用いて撮影された高精細の大容量画像データを扱うことが必要となる．そこで，オンデマンド VPN システムを導入することにより，大容量データの送受信時の転送効率にどの程度の影響が発生するかを実験により評価した．実験では 2 拠点間で 150MByte の顕微鏡画像を転送した場合の転送速度を計測した．オンデマンド VPN の影響を計測するため，VPN 接続を行わない場合の通常通信によるダウ

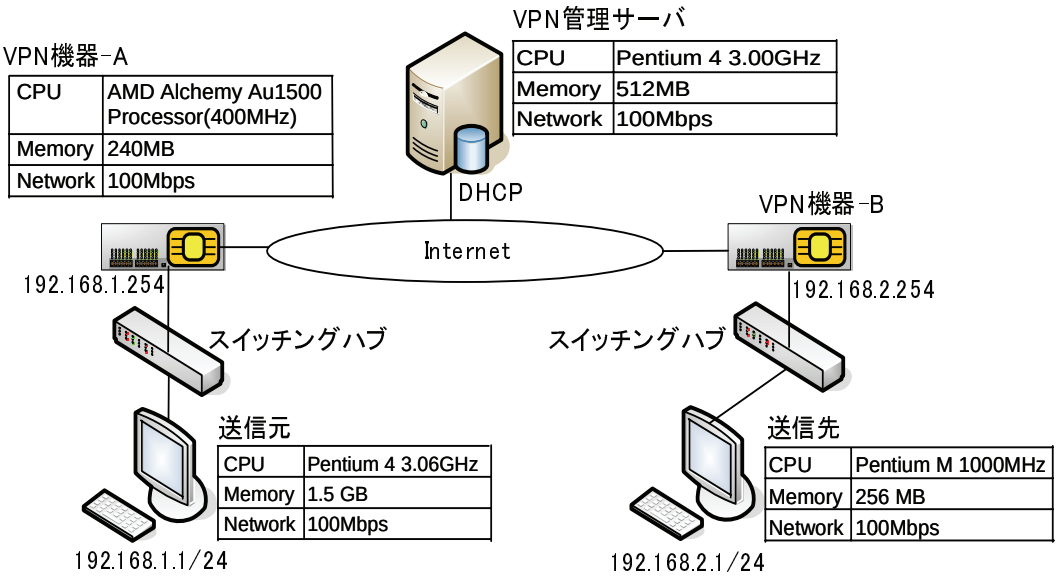

図 4.13 大容量画像データの転送実験環境

ンロード・アップロードに要する時間，VPN 接続を行った場合のダウンロード・アップロードに要する時間をそれぞれ測定した．実験環境を図 4.13 に，実験結果を図 4.14 にそれぞれ示す．

実験結果よりオンデマンド VPN を利用せずにデータを転送するために要した時間と，オンデマンド VPN を利用してデータを転送するのに要した時間はほぼ同一であることを確認することができた．従って，オンデマンド VPN システム利用の有無はデータの転送効率に殆ど影響を与えないといえる．今回の実験により，オンデマンド VPN システム導入によるデータ転送効率の低下は無視できるほど小さく，大容量データの送受信を行う医療現場への導入に問題がないことを確認することができた．

4.5.3　アクセス制御ポリシー可否判定時間の評価

本項では，オンデマンド VPN システムにおけるアクセス制御ポリシーの可否判定時間の評価方法とその結果・考察について述べる．アクセス制御ポリシーの可否判定時間を測定するため，登録されたポリシーの数に応じて利用フェーズにおける接続可否判定に要する時間がどのように変化するかを評価した．なお，アクセス制御ポリシーを定義するための専用のアプリケーション（図 4.15 参照）を開発し，GUI を用いて簡単に XACML

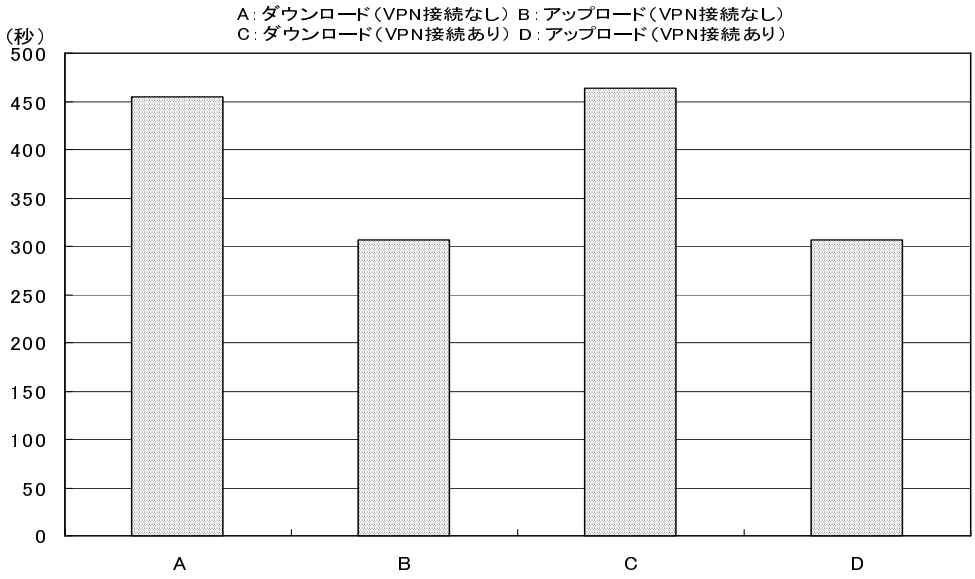

図 4.14 大容量画像データの転送実験結果

ポリシーを VPN 管理サーバに登録できるようにした．ここで，設定できるアクセス制御ポリシーは，認可ポリシーのみに限定している．

VPN 管理サーバに同数の接続元ポリシーと接続先ポリシーを登録し，VPN 管理サーバが接続要求を受信してから可否判定が完了するまでの時間を計測した．実験環境は図 4.13 と同一である．VPN 管理サーバでは 1 つのポリシーを 1 つのファイルとして管理している．今回の評価では，1 つのポリシーの中に，単一の主体，客体，動作の情報をそれぞれ記述し，可否判定に要する時間を計測した．計測結果を図 4.16 に示す．図 4.16 において，横軸のポリシー数は受信用アクセス制御ポリシーと発信用アクセス制御ポリシーの合計数を示している．

実験では，VPN 管理サーバが VPN 接続要求を受信してから接続可否判定が完了するまで，ポリシー 1 つ当たり約 0.01 秒要することを確認した．つまり接続制御を行うべき接続元・接続先の情報が合計で 100 個程度の小規模環境である場合には，可否判定に要する時間は 1 秒程度であり，十分に実用的であると考えることができる．一方で，登録されるポリシーの数が数千個になる大規模環境では，可否判定に数十秒必要であり，可否判定処理の性能向上が課題となる．

今回の実装では，可否判定エンジンの制約により，1 ポリシーを 1 ファイルとして管理する方式とした．エンジンの改良を行い，複数ポリシーを単一ファイルで管理できるよ

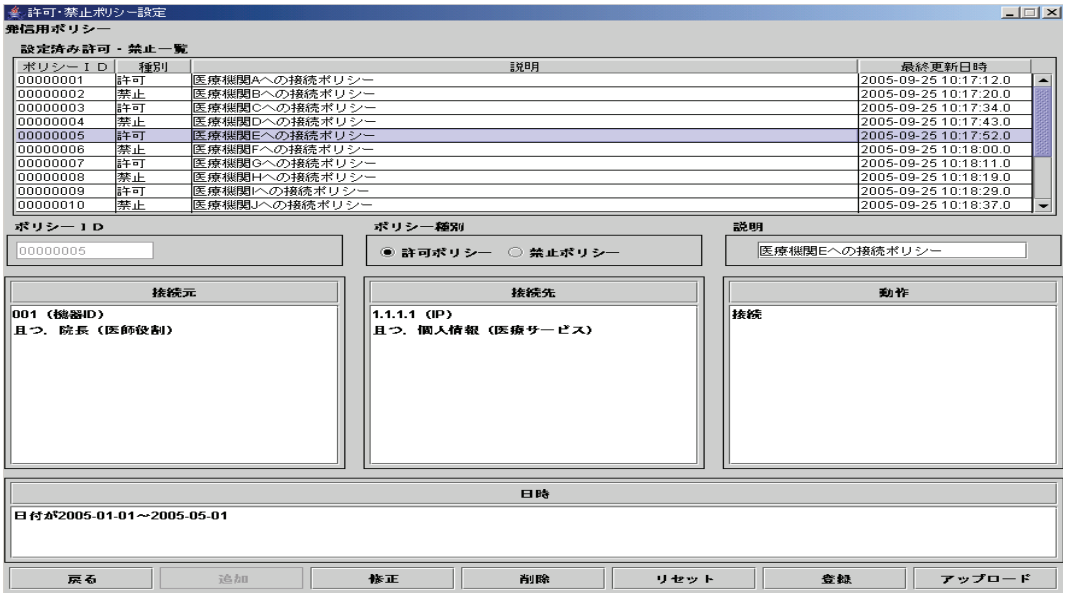

図 4.15 アクセス制御ポリシー登録 GUI

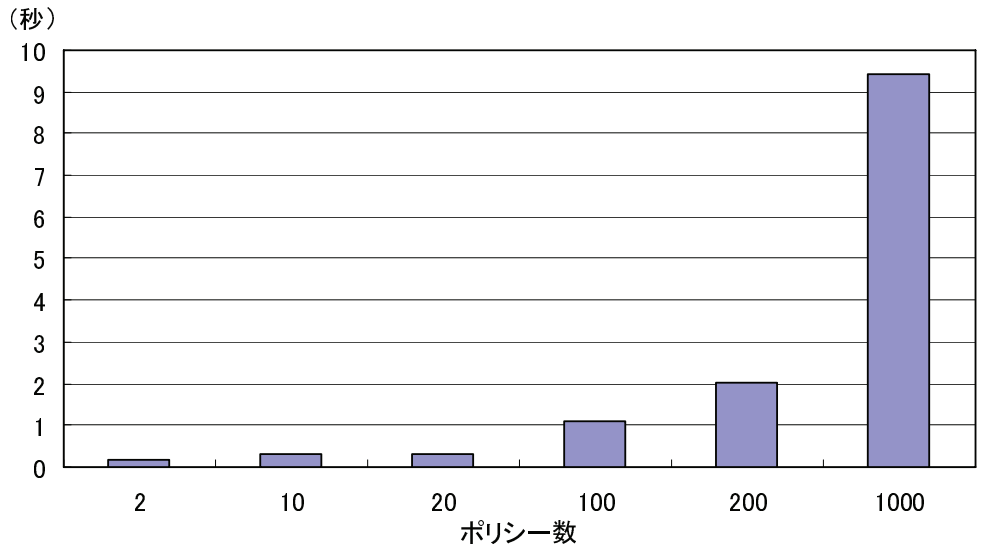

図 4.16 アクセス接続ポリシー可否判定時間

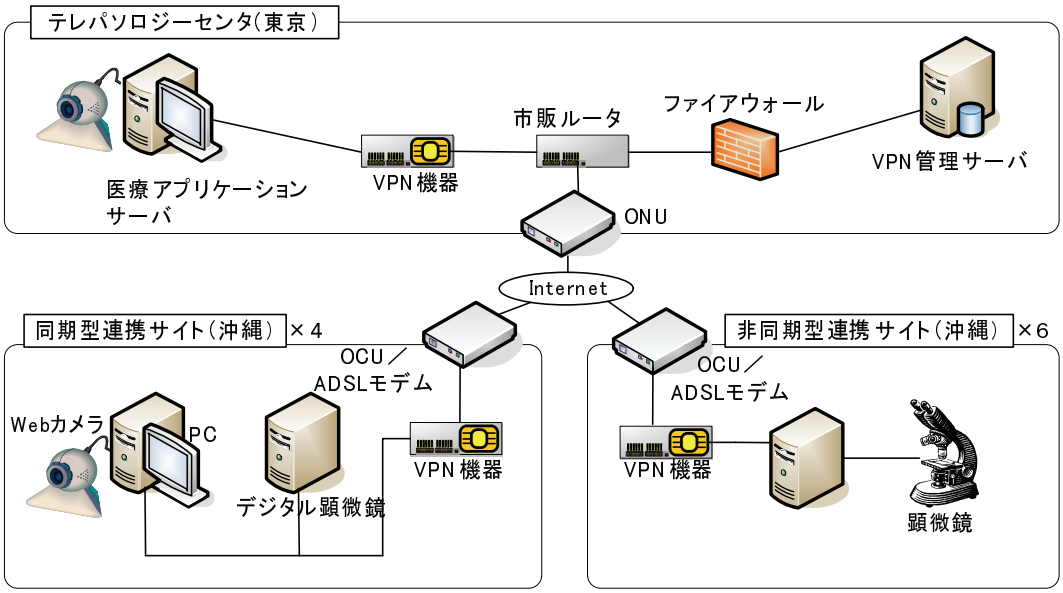

図 4.17 実証実験ネットワーク構成

う改善することにより，判定時間を短縮することができると考えられる．

4.5.4　フィールド評価

開発したプロトタイプを沖縄県の 10 カ所の医療機関に実際に導入し，テレパソロジーシステム[48]を用いた遠隔診療を行う実証実験を実施した．本項では，実環境でオンデマンド VPN システムを利用するための運用上の課題について考察する．

実証実験概要

医療分野では近年テレパソロジーシステムが導入されている．テレパソロジーシステムは通信ネットワークを通じて体組織の画像や顕微鏡の映像を送受信し，遠隔地の病理医が診断を下せるようにする遠隔病理診断システムである．テレパソロジーの運用形態には同期接続型と非同期接続型の 2 種類ある．同期接続型は，2 地点でリアルタイムに画像データなどをやり取りして遠隔診断を実施する形態である．非同期接続型とは，遠隔診断に必要なデータを一端サーバ上に保管し，そのデータを別の医師が読み取ることで診断を行う形態である．

テレパソロジーシステムで送受信されるデータは患者の診断データなどであり，その

運用には高度の安全性が求められる．そのため，テレパソロジーシステムを導入している医療機関同士には専用線を利用した VPN が開設されていることが一般的である．しかしながら，専用線の維持コストは高く，地方などの小規模医療機関でのテレパソロジーシステム導入の大きな障壁となっている．一方でオンデマンド VPN システムは，一般のインターネット網を利用してセキュアな通信路を確保することが可能であり，コストを低く抑えることが可能である．そこで，考案したオンデマンド VPN システムを利用することにより，低コストで安全にテレパソロジーシステムが利用できることを確認するために，実際の医療機関にて実証実験を実施した．

今回の実証実験では，沖縄県の 10 カ所の医療機関にオンデマンド VPN 機器を設置し，テレパソロジーシステムをオンデマンド VPN システムを通じて利用するための課題を検証した．VPN 管理サーバ・機器管理サーバ・非同期接続用テレパソロジーシステムサーバは東京に設置し実験を実施した．実証実験のネットワーク構成を図 4.17 に示す．各医療機関及びセンタは，インターネット網に一般の光ファイバ回線・ADSL 回線を経由して接続されている．

実証実験評価結果

実証実験では，インターネット回線の開設から VPN 機器の設置作業までのシステム設置に関わる評価と，VPN 機器の設定からオンデマンド VPN システム上でのアプリケーション利用までのシステム運用に関する評価を実施した．前者の評価では，事前準備フェーズに必要な作業を実環境において短期間かつ容易に実施可能であるかを中心に評価した．後者の評価では，利用フェーズに必要な作業をエンドユーザが容易に実施可能であるかを中心に評価した．評価方式はアンケート及びヒアリング調査による評価であり，実運用上のオンデマンド VPN システムに関する課題を明確にした．以下に示す (1)〜(4) が事前準備フェーズで必要となった作業と明らかになった課題であり，(5)〜(7) が利用フェーズで必要となった作業と明らかになった課題である．

(1) ［インターネット回線開設］：テレパソロジーシステムで送受信されるデータは画像データが主であり，大容量のインターネット回線が必要となる．従って実証実験では 10 の医療機関すべてに光ファイバ回線を開設することを目標とした．しかしながら，局舎との距離などの制約により光回線の開設が可能な医療機関は 2 カ所のみであり，8 カ所は ADSL 回線で代用した．申し込みから開設までの期間は ADSL が 1〜2 週間程度であったのに対し，光回線の開設には 1 ヶ月程度を要し

た．ユビキタス環境の利用促進に向け光回線インフラの普及と開設時間短縮の必要性が明らかとなった．

(2) ［利用施設内の回線敷設］：県立の医療機関では，敷設工事を行うために県の認可が必要となる．今回の実証実験では迅速に認可がおりず工事を開始できないケースが発生した．敷設工事に際し事前に関係機関と連携を密にする重要性が認識された．また屋内配線の工事が必要なケースもあり，利用者と施設担当者間の連携も必要となる．

(3) ［VPN 機器設置］：オンデマンド VPN ルータの設置作業は通常の通信機器の設置業務と異なる技術的技能や知識を要求されるものでない．アンケートでは 100% の利用者が，VPN 機器設置作業が容易であると回答した．

(4) ［VPN 機器設定作業］：今回の実証実験では VPN 機器登録作業・利用者登録作業を開発者が実施した．実際に利用者が作業を行うためには，手続きのマニュアル化が必要であると認識した．ただし登録作業は Web ベースで容易に行うことが可能であり，既存の IPsec 設定のように高度なネットワーク知識や機器の知識は不要である．

(5) ［VPN 接続開始］：VPN 接続開始要求は Web システムを用いて容易に行うことが可能である．アンケートでは 100% の利用者が VPN 接続のための作業は容易であると回答した．

(6) ［アプリケーション利用時］：VPN 接続完了後は，テレパソロジーアプリケーションから VPN 機能は遮蔽され意識する必要がない．20% の利用者から「本当に通信が暗号化されているか不安」という回答を得た．VPN 接続状況を可視化して表示することがユーザに安心感を与える意味で重要であると認識した．また，80% の利用者が VPN 接続の利用時間が 1 時間程度と回答した．

(7) ［VPN 切断］：アプリケーションと VPN システムは独立して機能しているため，アプリケーション終了時に VPN 切断を明示的に実施しないと VPN が接続されたままの状態になる．アンケートでは 20% の利用者がアプリケーションと VPN の接続・切断を同期させる必要があると指摘した．

評価の結果，オンデマンド VPN システム利用の前提となるインターネット回線の施設に長期間要するケースがあるものの，事前準備フェーズの作業自体は実環境でも容易に実施可能であることが明らかとなった．利用フェーズにおいては，GUI の改善が一部必

要であるものの，オンデマンド VPN サービスを利用する作業をエンドユーザが容易に実施可能であることが明らかとなった．以上により，オンデマンド VPN システムは実環境においても有効なシステムであると考えることができる．

4.5.5　評価のまとめと今後の課題

本項では，オンデマンド VPN システムのプロトタイプを用いて，4 つの観点から評価を実施した結果について述べた．VPN 構築に関する評価では，VPN 回線が構築されるまでに約 18 秒かかることを確認した．VPN 転送効率の評価では，オンデマンド VPN システムを利用することによるデータ転送効率への影響が端末 1 台の場合には無視できるほど小さいことを確認した．アクセス制御ポリシーの可否判定時間の評価では，小規模環境において適切なアクセス制御が十分に短い時間で実現できていることを確認した．フィールド評価では，医療機関にオンデマンド VPN システムを導入することにより，実際の現場での利用が可能であることを確認した．

一方で，それぞれの評価でいくつかの課題があることも明らかになった．以下に評価により明らかになった課題を示す．

(1) **VPN 構築時間に関する課題：**IKE のリトライ回数と時間を調節することにより VPN 構築までの接続時間を短縮することを検討する必要がある．また VPN の利用形態に応じては現在の構築に要する時間を短縮する必要がある．
(2) **VPN 転送効率に関する課題：**複数端末が同時接続を行った場合やインターネットのトラフィック量が，オンデマンド VPN システムの転送効率にどのような影響を及ぼすかを調査する必要がある．
(3) **アクセス制御ポリシーの可否判定時間：**VPN 利用者が増加するにつれ，必要となる接続ポリシーの数が飛躍的に増加する可能性がある．従って現状のポリシー制御技術の可否判定処理能力を向上させることが必要である．また，現在の XACML では単独の接続要求を判定することのみが可能であり，他の VPN 接続状況に応じた動的な可否判定を行うことができない．オンデマンド VPN の利用者が増加するに従い回線が混雑し，QoS 制御が必要になると考えられる．そのため，他の接続状況に応じて可否判定の結果を動的に変更することを可能にするポリシーの記述方式について検討を行うことが必要である．
(4) **実証実験に関する課題：**今回の実証実験は実施期間が短く，必ずしも十分な評価結

果を得ることができていない．今後継続して実証実験を行い，オンデマンド VPN システムのメリット，デメリットと改善事項などを明らかにしていくことが必要である．

4.6　結言

本章では，オンデマンド VPN システムを適用事例として，アクセス制御ポリシーから，VPN の接続に必要となる IPsec 構成情報を生成するアクセス制御ポリシー生成技術に関して論じた．また，VPN 機器の物理的安全性を確保するために必要となる 2 階層 PKI 技術についても論じた．

2 階層 PKI 技術を実装した耐タンパ IC チップによる厳密な機器認証により安全に，さらに，アクセス制御ポリシー生成技術を用いることにより，複雑な IPsec の VPN 構成情報を設定する手間を省き迅速に，VPN 接続を実現できることを確認した．また実証実験を通じて，オンデマンド VPN システムが実環境でも利用可能であることを確認した．

一方で，今回開発したプロトタイプを用いた評価では，接続時間の短縮やポリシー制御技術に関する技術上の課題と，実環境での運用上の課題を明確にした．今後これらの課題を解決する方式について検討を進めていく必要がある．

第 5 章

システム要件とアクセス制御ポリシーの整合性検証技術

5.1 緒言

第 3 章では，アクセス制御ポリシー内に含まれる矛盾や冗長性を検出する方式について論じ，第 4 章ではアクセス制御ポリシーに従って，IT システムが適切に動作するように，システムの設定情報を自動的に生成するアクセス制御ポリシー生成技術について述べた．これらの技術により，アクセス制御ポリシーが整合的で，かつアクセス制御ポリシーの設定通りにシステムが動作することを保証できるようになった．

しかしながら，これらの技術では，アクセス制御ポリシーの内容が，管理者が本来意図するシステム要件を満足しているかどうかを保証することはできない．例えば，個々に設定されたアクセス制御ポリシーには間違いがないが，アクセス制御ポリシーに漏れがあり，本来システムとしては到達すべきではない状態に到達したり，あるいは逆に，到達すべき状態に到達できないといった問題が発生するリスクがある．これは，一般にシステム管理者の意図するシステム要件が，非常に複雑な要件であるにも関わらず，アクセス制御モデルに，システム要件を本来の形式で記述できるほどの柔軟性が無いことが原因である．そのため，システム管理者は，本来のシステム要件を制約された形式に従って，システムに設定する必要がある．その結果，システム要件を誤って設定したり，あるいは，設定するべき要件を漏らすなどの問題が発生する．

この問題を解決するためには，設定されたアクセス制御ポリシーに従って，システムが

動作する場合に，その動作が本来のシステム要件を満足していることを検証する技術が必要となる．そこで本章では，モデル検査技術を用いて，システムがアクセス制御ポリシーに従ってどのような振る舞いをしたとしても，本来のシステム要件を満足することを検証する技術について論じる．

5.2　システム要件とアクセス制御ポリシーの整合性検証に関する技術課題

これまで論じてきたように，企業においては，内部犯行などによる機密情報漏洩が問題となっており，経営者にはそのリスクを極小化する努力が求められている．さらに，近年では金融商品取引法の改正（所謂，日本版 SOX 法）に伴い，経営者には努力義務のみならず，内部統制が有効に機能し，内部犯行による情報漏洩や改ざんのリスクに対して，十分な対策を施していることを，内部統制報告書の中で証明することが求められる．このような社会的要請の変化を背景に，企業においては，様々なセキュリティ管理製品の導入が進められている．

例えば，LDAP (Lightweight Directory Access Protocol) サーバのようなユーザ ID を管理するユーザ情報管理サーバ，共有ディレクトリへのアクセス権を制御するファイルサーバ，Microsoft Windows®Rights Management Services[36]や Adobe®LiveCycle™ Policy Server[46]のようにファイル単位で編集や印刷の可否を制御できる DRM (Digital Rights Management) 技術，印刷を制限するプリンタシステムなどが普及している．これらの技術を適切に組み合わせて機密情報の一貫したライフサイクル管理システムを構築することができれば，情報漏洩や改ざんのリスクを低減することが可能となる．しかしながら，実際にはすべてのシステムを適切に設定することは難しく，あるユーザが本来権限の無い動作をできるという機密性や完全性の問題，逆に権限のあるユーザが本来できるべき動作を妨げられるという可用性の問題が発生する可能性がある．特に，管理対象となるユーザの数が多く，システムの分散化・複雑化が進むと，各システム自身に設定されたアクセス制御ポリシーの整合性を検証することはできても，連携システム全体として，組織の機密性や可用性の要件を満足していることを証明することは困難である．

複数の制御装置にまたがるアクセス制御ポリシーが本来のシステム要件を満足していることを検証する研究としては，例えば VLAN の設定検証技術[41]やホームネットワークの連携サービス検証技術[56]がある．VLAN の設定検証技術では複数のスイッチに設定された内容が適切であり，届くべきではない相手にパケットが配信されたり，逆に届くべき相手にパケットが配信されないなどの不整合が発生しないことをモデル検査ツールである SMV (Symbolic Model Verifier)[35]を用いて検証する方法について提案している．また，ホームネットワークの連携サービス検証技術ではエアコンや温度計，煙探知機などの

家庭で使用されるネットワーク機器を協調させた連携サービスの設定情報を同様に SPIN を用いて検証する手法に関して提案している．しかしながら情報のライフサイクルに着目し，システムの設定が機密性や可用性を確保していることを検証する技術に関してはこれまで提案されていない．

そこで，本章では，あらゆる組織において必要となるドキュメント管理のセキュリティ問題を事例として，各システムに設定されたアクセス制御ポリシーが，システム要件である機密性や可用性の要件を満足しているかどうかをモデル検査技術[11] を用いて検証する，システム要件とアクセス制御ポリシーの整合性検証技術に関して論じる．

本章の構成は以下の通りである．第 5.3 節では，本章で対象とするドキュメント管理システムのモデルについて述べる．第 5.4 節では，ドキュメント管理システムに設定されたアクセス制御ポリシーと，システム要件との間に発生する整合性の問題について述べる．第 5.5 節では，システム要件とアクセス制御ポリシーの整合性を検証するモデル検査技術の概要について述べる．第 5.6 節では，対象とするドキュメント管理システムをどのようにモデル検査ツールを用いて検証するかについて述べる．そして第 5.7 節では，検証結果について述べ，最後に第 5.8 節でまとめを述べる．

5.3 ドキュメント管理システム

本節では，管理対象ドキュメント及びドキュメント管理システムの概要について述べる．ここではモデル検査技術を用いた汎用的なシステム要件とアクセス制御ポリシーの整合性検証技術を対象とするため，具体的な製品に特化したシステムではなく，一般的な特徴を兼ね備えたいくつかのシステムから構成される仮想ドキュメント管理システムを扱う．

5.3.1 管理対象ドキュメント

管理対象となるドキュメントは一般的な電子文書であり，読み込み・書き込み・印刷などの文書に関する動作が可能であるものとする．ただし，ドキュメントに対する各動作は以下で説明する種々のシステムによりユーザ単位または主体ロール単位で制限されるものとする．例えば，同一のドキュメントであってもユーザによっては読み込みはできるが，書き込みはできないという場合が考えられる．

5.3.2 ドキュメント管理システム

本章で対象とするドキュメント管理システムは，ユーザ情報管理システム，共有ディレクトリシステム，ファイル保護システムの 3 つのシステムから構成される．各システムの概要を以下に述べる．

ユーザ情報管理システム

ユーザ情報管理システムは，ユーザ ID 及びユーザが所属する主体ロールを管理する機能を提供する．具体的な製品の例としては LDAP サーバや Microsoft 社の提供する Active Directory などが該当する．

ユーザ情報管理システム上にはドキュメント管理システムを利用する組織内のすべてのユーザに 1 対 1 に紐付くユーザ ID が登録されている．簡単のため全ユーザの集合を U，個別のユーザ ID を $\{u_1, u_2, \cdots, u_n\}$ と表現することとする．さらに，ユーザ情報管理システムは複数の任意のユーザ ID をまとめた主体ロールを複数定義することができるものとする．すべてのユーザ ID は複数の主体ロールに所属することが可能である．第 3 章

と同様に，主体ロールを記号 S を用いて $S_1, S_2, \cdots$ などと表記する．

共有ディレクトリシステム

共有ディレクトリシステムは，管理対象ドキュメントを保存し，組織内ユーザがネットワーク経由で共有することのできる機能を提供する．ユーザ情報管理システムが管理するユーザ ID 及び主体ロール単位でディレクトリ内のドキュメントへの動作を制御することが可能である．

共有ディレクトリシステムはディレクトリごとに共有ディレクトリアクセス制御ポリシーを定義することが可能である．共有ディレクトリアクセス制御ポリシーは，第 3 章で定義した認可ポリシーと同様に，主体となるユーザ情報，客体となるディレクトリ情報，及び許可する動作の組から構成され

$$\texttt{Auth+}(U,\ D,\ A) \in \mathcal{P}.$$

と表現される．ここで，U はユーザ ID または主体ロール名，D はディレクトリ名，A は許可する動作名をそれぞれ表現するものとする．共有ディレクトリアクセス制御ポリシーで定義される動作の種類はディレクトリ内に保存されたファイルに対して閲覧を許可する read と，書き込みを許可する write の 2 種類が定義可能であるとする．

例えば，

$$\texttt{Auth+}(u_1,\ pass1,\ read).$$

は，pass1 という名称のディレクトリに対して，ユーザ u_1 は read が許可されるというアクセス制御ポリシーになる．また，

$$\texttt{Auth+}(S_1,\ pass2,\ write).$$

は，pass2 という名称のディレクトリに対して，主体ロール S_1 に所属するユーザは write が許可されるというアクセス制御ポリシーになる．

ファイル保護システム

ファイル保護システムは，Windows Rights Management Services や Adobe LiveCycle Policy Server のようなファイル保護技術を提供するシステムであり，ドキュメントごと

に読み込み・書き込み・印刷などの可否を制御することができる機能を提供する．実現方式にはいくつかの種類があるが，本章ではファイル保護情報をドキュメントに埋め込むことによりファイル保護を実現する方式を想定する．具体的には，ドキュメントを作成した時点でドキュメント作成者が，そのドキュメントに付与するファイル保護アクセス制御ポリシーを決定しドキュメントにポリシーを埋め込んで保存する．ファイル保護アクセス制御ポリシーはユーザ情報と動作の組から構成され

$$\texttt{Auth+}(U,\ *,\ A) \in \mathcal{P}.$$

と表現される．ここで，U はユーザ ID または主体ロール名，A は許可する動作名をそれぞれ表す．ファイル保護アクセス制御ポリシーは，客体となるドキュメント自身に定義されるため，客体の情報は記述する必要がなくなる．ファイル保護アクセス制御ポリシーで定義される動作の種類は当該ファイルの閲覧を許可する read，書き込みを許可する write，ファイルの内容を他のドキュメントへコピー&ペーストすることを許可する copy，印刷を許可する print の 4 種類が定義可能であるとする．

5.4　システム要件とアクセス制御ポリシーの関係

前節で定義したドキュメント管理システムにおいて，ユーザ情報の登録やディレクトリアクセス制御ポリシー，及びファイル保護アクセス制御ポリシーの設定を不適切に行うと，個々のシステムの設定に矛盾が無くても，本来システム全体として満足するべき要件を満足しない可能性がある．この問題を**システム要件とアクセス制御ポリシーの不整合**と呼ぶ．システム要件とアクセス制御ポリシーの不整合には，次に述べる機密性と，可用性に関する 2 種類の不整合が存在する．

5.4.1　機密性に関する不整合

ユーザがドキュメント作成時に付与したファイル保護アクセス制御ポリシー及び最初に保存したディレクトリに設定されている共有ディレクトリアクセス制御ポリシーにより許可されてない事象が発生することを，機密性に関する不整合と呼ぶ．

機密性に関する不整合は，厳密には次のように定義される．ドキュメントが最初に保存されたディレクトリのポリシーが Auth+(S_1, D, A_1)，ドキュメントに付与されたファイル保護アクセス制御ポリシーが Auth+$(S_2, *, A_2)$ であるとする．この時，何らかの過程を経て，あるユーザ $u \notin S_1 \cup S_2$ がある動作 $a \in A_1 \cup A_2$ をディレクトリ D に保存されたドキュメントに対して実行できる場合に，機密性に関する不整合が発生すると定義する．

5.4.2　可用性に関する不整合

ユーザがドキュメント作成時に付与したファイル保護アクセス制御ポリシー及び最初に保存したディレクトリに設定されている共有ディレクトリアクセス制御ポリシーにより許可されている動作ができない状態になることを，可用性に関する不整合と呼ぶ．

可用性に関する不整合は，厳密には次のように定義される．ドキュメントが最初に保存されたディレクトリのポリシーが Auth+(S_1, D, A_1)，ドキュメントに付与されたファイル保護アクセス制御ポリシーが Auth+$(S_2, *, A_2)$ であるとする．この時，何らかの過程を経てあるユーザ $u_1 \in S_1$，$u_2 \in S_2$ がある動作 $a_1 \in A_1$，$a_2 \in A_2$ をディレクトリ D に保存されたドキュメントに対して実行することが，それぞれ不可能な状態になる場合に，可用性に関する不整合が発生すると定義する．

5.5 モデル検査技術

ドキュメント管理システムに設定されたアクセス制御ポリシーにおいて，前節で定義した機密性及び可用性に関する不整合が発生するか否かをモデル検査技術を用いて検証する方式を述べる前に，本節ではモデル検査技術の仕組みと，モデル検査の実装であるモデル検査ツールの概要について述べる．

5.5.1 モデル検査の仕組み

モデル検査技術とは，形式手法の一つであり，有限状態遷移系でモデル化されたシステムの状態空間を網羅的に探索することにより，デッドロックや不活性などの望まない状態に到達する可能性があるか無いかを検証する技術である．具体的には，システムの有限状態遷移系を記述したものと，システムに要求される性質を時相論理式で記述したものを入力として与え，状態遷移系が時相論理式で与えられた性質を満足するかどうかを網羅的に探索する技術である．第 3 章で述べた検証技術は，第一階述語論理式を自由変数タブロー法を用いて検証する手法であったが，これは静的な検証を前提としていた．つまり，システム稼動後に変化する状態の変化を考慮した検証を行うことができない制約がある一方で，検証が有限時間で完了することが保証されたアルゴリズムであるという利点があった．それに対し，モデル検査技術は時相論理式を検証式として記述することが可能であるため，システム稼動後の主体や客体の状態変化を考慮した整合性の検証が可能になるという利点がある．一方で，状態の数が多くなると有限時間で検証が完了することを必ずしも保証できないという欠点がある．モデル検査技術の詳細については文献[11]に詳しく解説されている．

5.5.2 モデル検査ツール

モデル検査ツールとは，モデル検査技術に基づくアルゴリズムが実装されたソフトウェアであり，SPIN や LTSA[32]，SMV[35]などいくつかのツールが存在している．本章ではシステム間のメッセージ交換をモデル化した状態遷移系の記述の簡便性，及び記述可能な時相論理式の自由度が高いという観点で SPIN を用いた不整合の検証方式について述べる．

5.6　ドキュメント管理システムのモデル化

第 5.3 節及び第 5.4 節で定義したドキュメント管理システムのアクセス制御ポリシーとその不整合を SPIN を用いて検証するためには，各システムのアクセス制御ポリシーを SPIN の仕様記述言語である Promela[19]に変換する必要がある．本節では各ポリシーをどのように Promela に変換するかについて述べる．なお，本章で説明する Promela の完全なソースコードを第 5.6.2 項の末尾に掲載する．

5.6.1　アクセス制御ポリシーの事例

本項では，SPIN による検証の対象とする具体的なポリシーについて説明する．

最初に，ユーザ情報管理システムに登録されるユーザは $\{u_1, u_2, u_3, u_4\}$ の 4 ユーザであるとする．そして，2 つの主体ロール S_1, S_2 があり，$S_1 = \{u_1, u_2\}$，$S_2 = \{u_2, u_3, u_4\}$ という所属関係にあるとする．

次に，共有ディレクトリシステムには 2 つのディレクトリ D_A, D_B があり，以下の共有ディレクトリアクセス制御ポリシーが定義されているものとする．

$$\begin{array}{ll} \texttt{policy 1} & : \ \texttt{Auth+}(S_1,\ D_A, \{read, write\}) \\ \texttt{policy 2} & : \ \texttt{Auth+}(S_2,\ D_B, \{read, write\}) \end{array}$$

即ち，ディレクトリ D_A には，主体ロール S_1 に所属するユーザが read または write 可能であり，ディレクトリ D_B には，主体ロール S_2 に所属するユーザが read または write 可能であるというポリシーが定義されている．

また，今回はユーザ u_1 がドキュメントの作成者であり，作成したドキュメントをディレクトリ D_A に保存する際に，下記のファイル保護アクセス制御ポリシーを当該ドキュメントに設定したと仮定する．

$$\texttt{policy 3} \ : \ \texttt{Auth+}(S_1,\ *,\ print)$$

即ち，u_1 が作成したドキュメントは主体ロール S_1 に所属するユーザが印刷可能である．

これらのユーザ情報及びアクセス制御ポリシーをどのように Promela で記述するかについて次項で説明する．

5.6.2 Promela による記述

ユーザ情報管理システムで管理されるユーザ ID は mtype 型のデータとして下記のように最初に宣言する．

```
mtype={u1, u2, u3, u4}
```

そして各ユーザがどの主体ロールに属しているかは，Promela のマクロ記述である inline 文を用いて必要な場合に判定するようにする．具体的には下記に示すように，ユーザが u_1 または u_2 である場合には，主体ロール S_1 に所属していることを示す変数 u_in_s1 を true にすることにより，ユーザの所属を判定する．

```
inline s1(u){
  if
    ::(u==u1||u==u2)->u_in_s1 = true
    ::else->skip
  fi}
```

次に共有ディレクトリアクセス制御ポリシーとファイル保護アクセス制御ポリシーについて述べる．共有ディレクトリシステムでは，各ディレクトリ毎にポリシーが定義されており，ユーザ情報に応じて実行可能な動作が異なる．そこで 2 重の if 文を用いて下記のような記述でポリシーを表現する．

```
if
 :: (file==directory_a) ->
 if
  ::u_in_s1->event=read; file=directory_a
  ::u_in_s1->event=write;file=directory_a
  ::u_in_s1->event=print;file=directory_a
  ::(u_in_s1 && u_in_s2)
           ->event=write;file=directory_b
  :: else -> skip
 fi;
 ::else -> skip
fi
```

最初の if 文では現在の操作対象のドキュメントがどのディレクトリに保存されているかの判定を行う．上記の記述ではディレクトリ D_A に関する処理について記述している．二つめの if 文では共有ディレクトリアクセス制御ポリシーとファイル保護アクセス制御ポリシーで設定されている制約を記述する．即ち，$u \in S_1$ である場合には，read, write, print のいずれの動作も可能であり，動作後も該当ドキュメントはディレクトリ D_A に保存され続ける．しかし，$u \in S_1$ かつ $u \in S_2$ に所属するユーザが存在する場合には，当該ファイルをディレクトリ D_B に保存する可能性があることを示している．ディレクトリ D_B に関しても同様の記述となる．

最後に，ドキュメントを操作するユーザが非決定的に選択されるように Promela で記述することにより，モデル化が完了する．実際のシステムにおいてはどのユーザがどの動作を行うかを事前に把握することは不可能である．そこで下記のように，ユーザが非決定的に選択されるようにする．

```
if
 :: true -> u=u1 ; s1(u1) ; s2(u1)
 :: true -> u=u2 ; s1(u2) ; s2(u2)
 :: true -> u=u3 ; s1(u3) ; s2(u3)
 :: true -> u=u4 ; s1(u4) ; s2(u4)
fi;
```

本項の最後に，次節の検証で用いる Promela のソースコードを掲載する．

Promela のソースコード

```
01: mtype = {u1, u2, u3, u4};
02: bool u_in_s1;
03: bool u_in_s2;
04: mtype = {directory_a, directory_b};
05: mtype = {create, read, write, print};
06: mtype file;
07: mtype event;
08: mtype u;

10: inline s1(u){
11:  if
12:     ::(u==u1 || u==u2) ->
13:        u_in_s1 = true
14:     ::else -> skip
15:   fi
16: }

17: inline s2(u)
18: {
19:   atomic{
20:   if
21:     ::(u==u2 || u==u3 || u==u4) ->
22:        u_in_s2 = true
23:     ::else -> skip
24:   fi
25:   }
26: }
```

```
28: active proctype policy(){
29: file = directory_a;
30: event = create;
31: u=u1;
32: do
33:  ::true ->
34:  atomic{
35:  if
36:   :: true -> u=u1 ; s1(u1) ; s2(u1)
37:   :: true -> u=u2 ; s1(u2) ; s2(u2)
38:   :: true -> u=u3 ; s1(u3) ; s2(u3)
39:   :: true -> u=u4 ; s1(u4) ; s2(u4)
40:  fi;
41:  if
42:   :: (file==directory_a) ->
43:   if
44:    :: (u_in_s1==true) -> event=read;
45:                      file=directory_a
46:    :: (u_in_s1==true) -> event=write;
47:                      file=directory_a
48:    :: (u_in_s1==true) -> event=print;
49:                      file=directory_a
50:    :: (u_in_s1==true && u_in_s2==true) ->
51:                      event=write;
52:                      file=directory_b
53:    :: else -> skip
54:   fi;
55:   ::else -> skip
56:  fi;
57:  if
58:   :: (file==directory_b) ->
```

```
59:   if
60:    :: (u_in_s2==true) -> event=read ;
61:                       file=directory_b
62:    :: (u_in_s2==true) -> event=write;
63:                       file=directory_b
64:    :: (u_in_s1==true && u_in_s2==true) ->
65:                       event=write;
66:                       file=directory_a
67     :: else -> skip
68    fi;
69    ::else -> skip
70   fi;
71  }
72  od
73: }
```

5.7　システム要件とアクセス制御ポリシーの整合性検証

本節では，第 5.6 節で説明した Promela 記述を用いて，機密性と可用性の不整合を SPIN により検証し得ることを示す．

5.7.1　機密性の検証

機密性の不整合の例として次のような事例を考える．

- u_1 が S_1 に所属するユーザのみに閲覧・印刷させる目的で作成したドキュメントを S_2 に所属する u_1 以外のユーザが閲覧できる．

このことを検証するためには，以下の式を検証式として SPIN に入力し反例の有無を検証すればよい．

$$\Box\neg((u \notin S_1) \wedge (event = read) \wedge (u \in S_2))$$

ここで，記号 □ は時相論理で使用される記号であり，$\Box A$ とした場合，性質 A が常に成立し続けることを意味する記号である．即ち，主体ロール S_1 に所属せず，主体ロール S_2 に所属するユーザが当該ドキュメントを read できるという状態が決して発生しないことを検証する．SPIN では与えられた検証式に対して，それを満足するかどうかを，全ての状態を網羅的に探索することにより検証する．探索の結果，条件を満足する場合には，“valid”という出力結果を返し検証が完了する．条件を満足しない場合には，“invalid”という結果を返すと同時に，条件を満足しない状態に至るまでの，状態遷移を反例としてログ出力する．このログのことを反例トレースと呼ぶ．

SPIN において上記の時相論理式を入力する場合には，以下のように記述する．

```
[]   ! (p && q && r)
#define p u!=u1
#define q event==read
#define r u_in_s2==true
```

前述のモデルに対してこの検証式を検証すると反例が出力される．従って現状のポリシーでは機密性が保証されない可能性があることがわかる．出力の一部を図 5.1 に示す．

SPIN の出力する反例トレースを詳細に検証すると，次の場合に機密性の不整合が発生することがわかる．つまり u_2 が主体ロール S_1, S_2 の両方に所属しているため，u_2 がディレクトリ D_A に保存された当該ドキュメントをディレクトリ B_2 に移動した場合に，主体ロール S_2 に属する u_2 以外のユーザが閲覧でることになる．

第 3 章述べた自由変数タブロー法による検証手法は第一階述語論理式を前提としているため，状態の変化を考慮しない静的な解析が対象であった．一方でモデル検査技術は，時相論理式を前提として検証することが可能であるため，この例のように，主体や客体が動的に変化してはじめて発生する不整合を検証可能であるという利点がある．

5.7.2　可用性の検証

可用性の不整合の例として次のような事例を考える．

- u_1 が S_1 のみに閲覧・印刷させる目的で作成したドキュメントが S_1 に所属するユーザであっても印刷できない状態が発生する．

このことを検証するためには，以下の式を検証式として SPIN に入力し，反例の有無を検証すればよい．

$$\Diamond((u \in S_1) \wedge (event = print))$$

ここで，記号 $\Diamond$ は時相論理で使用される記号であり，$\Diamond A$ とした場合，性質 A がいつか成立し得る状態に到達できることを意味する記号である．即ち，主体ロール S_1 に所属するユーザであれば，print という動作をいつか実行することができることを検証する．

SPIN において上記の時相論理式を入力する場合には，以下のように記述する．

```
<>  (p && q)
#define p u_in_s1==true
#define q event==print
```

前述のモデルに対してこの検証式を検証すると反例が出力される．従って現状のポリシーでは可用性が保証されない可能性があることがわかる．出力の一部を図 5.2 に示す．SPIN の出力する反例トレースを分析すると，u_2 が主体ロール S_1 と S_2 の両方に所属しているため，u_2 がディレクトリ D_A に保存された当該ドキュメントをディレクトリ D_B に

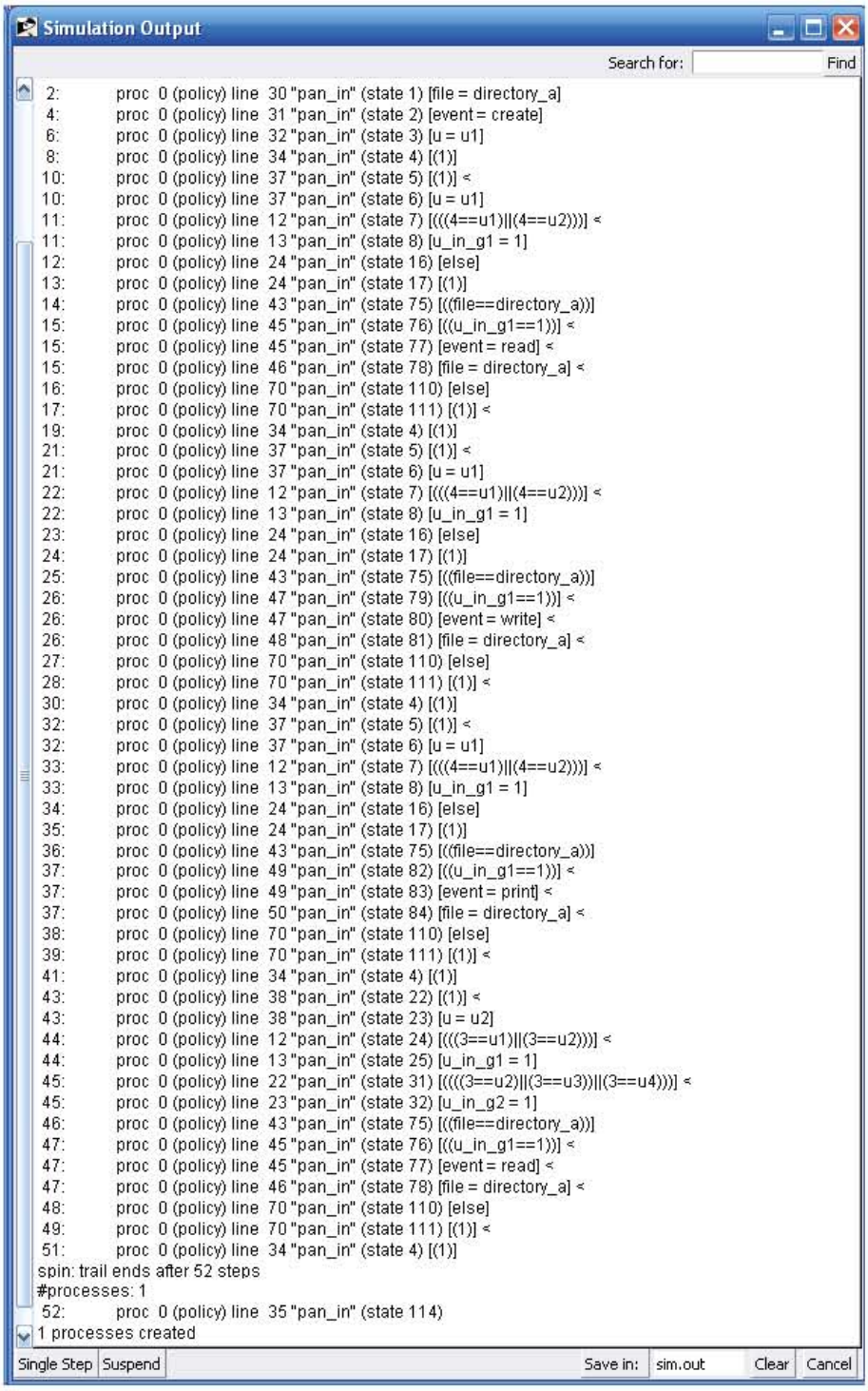

図 5.1 SPIN の反例トレース (機密性の不整合の検証結果)

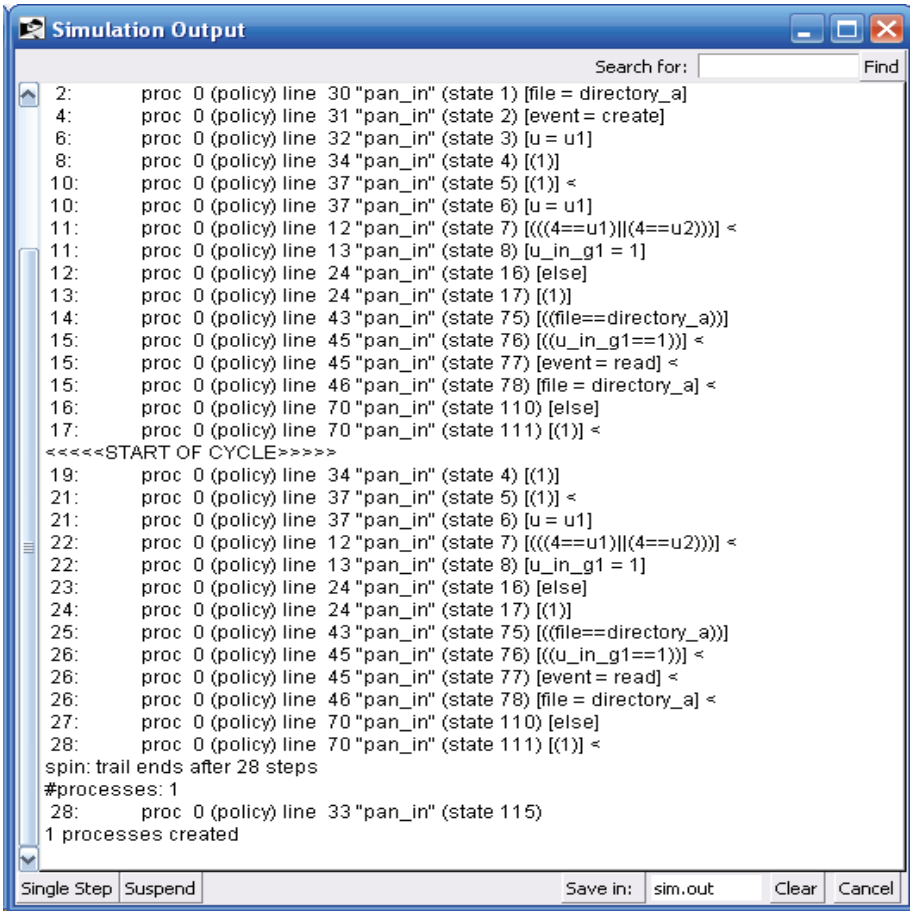

図 5.2 SPIN の反例トレース (可用性の不整合の検証結果)

移動した場合に，主体ロール S_1 に属するユーザが印刷することができる可能性があることがわかる．

以上のように，モデル化したドキュメント管理システムをモデル検査技術を用いて検証することにより，機密性と可用性の不整合を検出可能であることを確認した．

5.7.3　検証時間とモデル検査技術の限界に関する考察

最後にモデル検査技術による検証時間と，その限界について考察する．

今回のモデルケースによる検証では，上記の機密性に関する不整合と，可用性に関する不整合を検証するまでに要した時間は，それぞれ 0.015 秒であることを SPIN のログにより確認した．このことから，検証に殆ど時間を要していないことがわかる．これは，設定

したユーザの数とアクセス制御ポリシーの数が少ないためであると推測できる．

SPIN の検索アルゴリズムは高度に最適化されているため，ある程度大きな状態遷移モデルであっても検証を行うことが可能である．しかしながら，マシンのリソース上の制約から探索可能な状態遷移空間には制限があり，ユーザの数やアクセス制御ポリシーの数が極端に多くなると，網羅的な検証ができずに，“valid”でも “invalid”でもなく探索不能という形で検証が終了する可能性がある．どの程度の数のアクセス制御ポリシーを SPIN により解析可能であるかを明らかにすることは今後の課題である．

また，SPIN では与えられた時相論理式を満足しない反例を 1 つでも検出した場合，そこで検証を終了し反例トレースを出力する．このことは SPIN に限らず，LTSA や SMV などの他のモデル検査ツールも同様である．従ってモデル検査ツールによる検証では，仮に不整合が複数存在したとしても，常に一つの不整合のみしか検出できない制約がある．そのため実システムの検証においては，不整合を検出する都度，その原因を除去し，再度反例トレースが出力されなくなるまで，モデル検査ツールによる検証を繰り返す必要がある．

5.8 結言

本章では，ドキュメント管理システムのユーザ管理情報やポリシーを仕様記述言語の Promela で記述し，モデル検査ツールの SPIN を用いて解析することにより，機密性と可用性の不整合を検証する方式について提案した．また簡単な事例を用いて，実際に検証が可能であり，不整合が発生する場合には，どのよう手順で発生するかを確認することができることを示した．

今後は，対象とするシステムを拡大し，例えばルーターやメールサーバなどのネットワーク機器，プリンタなどの出力機器を含めた整合性検証技術に関して検討を行う予定である．また，各システムの設定情報から自動的に Promela 記述への変換を行う方式についても検討を行う予定である．

第 6 章

結論

近年，IT システムは，社会生活に急速に深く浸透し，日常生活に欠くことのできない社会インフラとしての役割を担いつつある．その反作用として，IT システムが原因の情報漏洩などのセキュリティインシデントや，システムトラブルによる金融機関や交通機関の業務停止などが社会問題化している．これらの原因は，IT システムが巨大化しその制御が高度に複雑化したために，これまで専門家の経験とスキルに頼っていた運用管理手法が限界に達したことにある．特に個人情報などの情報漏洩事故は公表されているだけでも相当数にのぼり，それが与える経済的損失及び個人が受ける精神的苦痛は大きな問題であり，早急な技術的対策が求められている．

そこで，本論文では，情報漏洩の原因の大部分を占めるアクセス制御の問題について，その技術課題を明らかにし，解決策について議論した上で，その有効性を考察した．本論文で議論した技術を適用することにより，一定の制約条件の下で，IT システムが管理者の設定したアクセス制御に従って正しく振る舞うことを検証することが可能となった．それにより，運用ミスによる機密情報漏洩などのセキュリティインシデントのリスクを大幅に低下させることを実現した．

以下に，本研究の個々の検討で得られた結果について示す．

第一に，IT システムにおけるアクセス制御ポリシーの重要性とアクセス制御ポリシーを用いて IT システムを制御するために必要となる技術とその課題について考察した．その上で，アクセス制御ポリシー生成技術とアクセス制御ポリシー検証技術が情報漏洩を防止する上で重要な技術であること述べ，本研究の意義を明確にした．

第二に，アクセス制御ポリシーを利用したシステムにおいて，矛盾するアクセス制御

ポリシーや冗長なアクセス制御ポリシーがシステムに与える悪影響を明らかにした．その上でそれらの問題あるポリシーを静的に検証することを可能にするアクセス制御ポリシー整合性検証技術について考察した．アクセス制御ポリシー整合性検証技術を用いることにより，ある一定の条件を持つアクセス制御モデルの中に含まれる単純矛盾，暗黙的な単純矛盾，制約矛盾の 3 種類の矛盾するポリシーと冗長なポリシーを検証可能であることを数学的に証明した．またシミュレーションにより，考案した技術により矛盾するアクセス制御ポリシーの検出がシステム上も可能であり，1000 個程度のアクセス制御ポリシーの解析を十分に短い時間で解析できることを明らかにした．

第三に，医療機関で利用されるオンデマンド VPN システムを事例として，アクセス制御ポリシーを VPN 接続を実現する複数のネットワーク機器が解釈可能な形式に変換し，リアルタイムかつ安全に配信するアクセス制御ポリシー生成技術について議論した．異なる医療機関を結ぶネットワークには，医療情報などの高い機密性が求められる情報が流れるため，ネットワーク経路の安全性は当然のことながら，万が一ネットワーク機器が盗難された場合にも備え，なりすましの問題にも対応できることが求められていた．そこで 2 階層 PKI 技術を用いた機器認証技術を用いることで，機器の盗難による問題を解決すると共に，アクセス制御ポリシー生成技術により，これまで実現が難しかった任意の二地点間をリアルタイムかつ安全に IPsec を用いた VPN で接続することを実現した．そして考案技術を搭載したネットワーク機器を用いて実証実験を行うことにより，迅速性が求められる医療機関においても十分に短い時間で VPN 接続可能であることを確認した．さらに，これまで複雑な設定が必要であった VPN 接続作業が，考案方式を用いることにより設定作業の負荷が大幅に軽減されると共に，安全性が高まったことを利用者アンケートにより明確にした．

第四に，ドキュメント管理システムを事例として，システム要件とアクセス制御ポリシー間の不整合を検証する手法について考察した．上記で述べたアクセス制御ポリシー整合性検証技術とアクセス制御ポリシー生成技術を用いることにより，IT システムに設定されたアクセス制御ポリシー自身に問題がなく，かつ，アクセス制御ポリシーの定義通りにシステムが振る舞うことを保証することが可能となった．しかしながら，これらの技術では，システムの振る舞いが本来システムが満足すべき要件を満たしていることを保証することはできなかった．例えば，個々に定義されたアクセス制御ポリシーが正しいとしても，その定義に漏れがあれば，本来権限のないユーザが機密情報にアクセスセスできるという機密性の不整合や，逆に本来権限のあるべきユーザが情報にアクセスでき

ないという可用性の不整合が発生する可能性がある．そこで，これらの不整合をモデル検査技術を用いて検証する手法を考案し，その有効性を検証した．その結果，あるモデルケースを前提とした実験により，システム要件とアクセス制御ポリシー間の不整合を検証可能であることを明らかにした．

次に本研究分野における今後の課題について述べる．

まず，第 3 章で述べたアクセス制御ポリシー整合性検証技術は，第一階述語論理式により表現されるアクセス制御ポリシーの検証を前提としている制約がある．このことは，検証が有限時間内に完了することを保証する利点がある一方で，動的に変化する主体や客体，システムの状態を踏まえた検証が不可能であるという欠点がある．実際にはシステム稼動後に，これらの状態は刻一刻と変化する．そのため，設定されたアクセス制御ポリシーを静的に解析した結果に問題が無かったとしても，状態変化に伴い，新たなポリシー矛盾が発生する可能性がある．そこで，これらの状態変化にも対応できるように，アクセス制御ポリシー整合性検証技術を拡張する必要がある．

そして，第 4 章で述べたアクセス制御ポリシー生成技術は，認可ポリシーのみを特定の IPsec 構成情報に変換する技術であるという制約がある．つまり，認可ポリシー以外の，強制ポリシーや制約ポリシー，あるいは，IPsec 以外の暗号化技術を用いた VPN 接続には対応していない．医療機関の多種多様性を考慮すると，これらの技術にも対応できるように考案方式を拡張する必要がある．

第 5 章で述べたシステム要件とアクセス制御ポリシーの整合性検証技術は，ユーザ管理システムと，共有ディレクトリアクセス制御ポリシー，及びファイル保護アクセス制御ポリシーを対象とした技術である．しかしながら，近年は IC カードによる認証を経てから利用可能となるプリンターや，USB メモリへの書き込みに管理者の承認を必要とするソフトウェアなど，オフィス内で利用されるセキュリティ製品は多岐にわたっている．情報漏洩を根絶するためには，これらのセキュリティ製品のアクセス制御ポリシーも対象にできるように，方式を拡張する必要がある．

最後に，本論文において検討したアクセス制御ポリシー生成技術とアクセス制御ポリシー整合性検証技術がより高度に発展し，情報漏洩の無い安全で安心な IT システムの未来が切り開かれることを願い，本論文を締めくくる．

謝辞

本論文は，大阪大学大学院工学研究科電気電子情報工学専攻教授 馬場口登博士のご指導のもと，筆者が株式会社エヌ・ティ・ティ・データ在職中及び，大阪大学大学院工学研究科電気電子情報工学専攻在学中に行った研究成果をまとめたものである．本研究を進めるにあたり，研究内容に関してのみならず，論文のまとめ方に至るまで，ご多忙の中，長期間にわたり常に丁寧なご指導とあたたかいご鞭撻をいただいた 馬場口登博士に，心より感謝申し上げる．また本論文を完成させるにあたっては，多数の有益なご指摘を大阪大学大学院工学研究科教授 滝根哲哉博士及び大阪大学産業科学研究所教授 溝口理一郎博士からいただいたこと，厚く感謝申し上げる．

また，筆者の大学院在学中，研究を進めていく過程において講義等を通じ，本論文に関して多大なるご指導，ご助言を賜った，大阪大学大学院工学研究科教授 河崎善一郎博士，同教授 北山研一博士，同教授 小牧省三博士，同教授 三瓶政一博士，同教授 井上恭博士，大阪大学産業科学研究所教授 鷲尾隆博士をはじめとする先生方に，厚く御礼申し上げる．

さらに，株式会社エヌ・ティ・ティ・データ技術開発本部副本部長 山本修一郎博士及び同情報セキュリティ推進室長 山岡正輝博士，ならびに日本システム監査法人 松田栄之氏をはじめとする株式会社エヌ・ティ・ティ・データ技術開発本部の皆様から，研究に関する議論を通じて数多くの気づきを与えていただいたことで，研究の方向性やアイディアを取りまとめていくことができたことに対し，深く感謝の意を表する．

本研究を進めるにあたり，インペリアル大学教授 Morris Sloman 博士，同上級講師 Krysia Broda 博士，国立情報学研究所教授 本位田真一博士，同特任教授 田口研治博士，電気通信大学大学院情報システム学研究科准教授 田原康之博士には，数多くのソフトウェア工学に関する最先端の手法に関して丁寧にご指導，ご助言を賜ったことに対し，厚く感謝申し上げる．

また，大阪大学の伊藤義道助教，博士課程学生の河野和宏氏をはじめとする馬場口研究

室の諸氏には，筆者が博士課程在籍中にお世話になった．記して感謝する．

本研究の一部は，総務省の平成 17 年度「高度ネットワーク認証基盤技術の研究開発」の委託を受けた「オンデマンド VPN 技術についての研究開発」に関するものである．関係者各位に感謝申し上げる．また実証実験に協力いただいた東京工業大学教授 大山永昭博士，同准教授 小尾高史博士，同助手 鈴木裕之博士，ハーバード大学の八木由香子助教に深く感謝申し上げる． さらに，業務が多忙の中，研究内容にご理解をいただき，長期にわたりご支援をいただいた，株式会社エヌ・ティ・ティ・データ代表取締役常務執行役員 山田伸一氏，同執行役員 山田英司氏，同技術開発本部副本部長 木谷強博士，エヌ・ティ・ティ・データ先端技術株式会社常務取締役事業部長 松本隆明博士をはじめとする諸兄姉に，改めて厚く御礼申し上げる．

参考文献

1) A. Bandara, E. Lupu, A. Russo, N. Dulay, M. Sloman, P. Flegkas, M. Charalambides, and G. Pavlou, "Policy Refinement for DiffServ Quality of Service Management," *The 9th IFIP/IEEE International Symposium on Integrated Network Management (IM2005)*, pp. 469–482, 2005.
2) B. Beckert and R. Goré, "Free Variable Tableaux for Propositional Modal Logics," *International Conference on Theorem Proving with Analytic Tableaux and Related Methods*, 1997.
3) B. Beckert and J. Posegga, "leanTAP: Lean Tableau-Based Deduction," *Journal of Automated Reasoning*, Vol. 15, No. 3, pp. 339–358, 1995.
4) E. Bertino, P. A. Bonatti, and E. Ferrari, "TRBAC: A Temporal Role-Based Access Control Model," *ACM Transactions on Information and System Security*, Vol. 4, No. 3, pp. 191–233, 2001.
5) R. Bhatti, J. B. D. Joshi, E. Bertino, and A. Ghafoor, "Access Control in Dynamic XML-Based Web-Services with X-RBAC," *The 2003 International Conference on Web Services (ICWS2003)*, pp. 243–249, 2003.
6) D. F. C. Brewer and M. J. Nash, "The Chinese Wall Security Policy," *IEEE Symposium on Security and Privacy*, pp. 206–214, 1989.
7) N. G. Carr, "IT Doesn't Matter," *Harvard Business Review*, Vol. 5, pp. 41–49, 2003.
8) N. G. Carr, "The End of Corporate Computing," *MIT Salon Management Review*, Vol. 46, No. 3, pp. 66–73, 2005.
9) L. Cholvy and F. Cuppens, "Analyzing Consistency of Security Policies," *IEEE Symposium on Security and Privacy*, pp. 103–112, 1997.
10) D. D. Clark and D. R. Wilson, "IEEE Symposium on Security and Privacy," pp. 184–19,

1978.

11) E. M. Clarke, O. Grumberg, and D. Peled, "Model Checking," *The Mit Press*, 2000.
12) N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The Ponder Policy Specification Language," *IEEE 2nd International Workshop on Policies for Distributed Systems and Networks (POLICY2001)*, pp. 18–39, 2001.
13) P. H. Deussen, I. Schieferdecker, and H. Kamoda, "A Methodology for Policy Conflict Detection Using Model Checking Techniques," *The 24th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE2004)*, pp. VII(1)–(14), 2004.
14) D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST Standard for Role-Based Access Control," *ACM Transactions on Information and System Security*, Vol. 4, No. 3, pp. 224–274, 2001.
15) M. Fitting, "First Order Logic and Automated Theorem Proving," *Springer*, 1996.
16) Z. Fu, S. F. Wu, H. Huang, K. Loh, F. Gong, I. Baldine, and C. Xu, "IPSec/VPN Security Policy: Correctness, Conflict Detection, and Resolution," *IEEE 2nd International Workshop on Policies for Distributed Systems and Networks (POLICY2001)*, pp. 36–43, 2001.
17) A. Graham, T. Radhakrishnan, and C. Grossner, "Incremental Validation of Policy-Based Systems," *IEEE 5th International Workshop on Policies for Distributed Systems and Networks (POLICY2004)*, pp. 240–249, 2004.
18) D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)," *Request for Comments 2409*, 1998.
19) G. J. Holzmann, "The SPIN Model Checker, Primer and reference manual," *Addison Wesley*, 2004.
20) N. Ichihara, H. Kamoda, and H. Oguro, "Smartcard Security Development Using Formal Method Tool SPIN," *The 9th International Common Criteria Conference (ICCC2008)*, pp. 2405(1)–(15), 2008.
21) S. Jajodia, P. Samarati, and V. S. Subrahmanian, "A Logical Language for Expressing Authorizations," *IEEE Symposium on Security and Privacy*, pp. 31–42, 1997.
22) A. A. E. Kalam, S. Benferhat, A. Miége, R. E. Baida, F. Cuppens, C. Saurel, P. Balbiani, Y. Deswarte, and G. Trouessin, "Organization Based Access Control," *IEEE 4th Inter-*

national Workshop on Policies for Distributed Systems and Networks (POLICY2003), pp. 120–131, 2003.

23) J. A. Kalman and L. Wos, "Automated Reasoning with Otter," *Rinton Press*, 2001.

24) H. Kamoda, "Access Control Policy Verification Using Model-Checking," *The 1st Joint Workshop on Security Software Engineering in Tokyo*, p. 12, 2007.

25) H. Kamoda, A. Hayakawa, M. Yamaoka, S. Matsuda, K. Broda, and M. Sloman, "Policy Conflict Analysis Using Tableaux for On Demand VPN Framework," *The 6th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM2005)*, pp. 565–569, 2005.

26) H. Kamoda, M. Yamaoka, S. Matsuda, K. Broda, and M.Sloman, "Policy Conflict Analysis Using Free Variable Tableaux for Access Control in Web Services Environments," *The 14th International World Wide Web Conference (PM4W2005)*, pp. 5–12, 2005.

27) H. Kamoda, M. Yamaoka, S. Matsuda, K. Broda, and M. Sloman, "Access Control Policy Analysis Using Free Variable Tableaux," *Transactions of Information Processing Society of Japan (IPSJ)*, Vol. 47, No. 5, pp. 1515–1529, 2006.

28) E. Letier, "Reasoning about Agents in Goal-Oriented Requirements Engineering," *Ph. D. Thesis, University of Louvain*, 2001.

29) R. Letz and G. Stenz, "Model Elimination and Connection Tableau Procedures," *Handbook of automated reasoning*, pp. 2015–2112, 2001.

30) Linux Free S/WAN Project, "FreeS/WAN documentation," 2003.

31) E. C. Lupu and M. Sloman, "Conflicts in Policy-Based Distributed Systems Management," *IEEE Transactions on Software Engineering*, Vol. 25, No. 6, pp. 852–869, 1999.

32) J. Magee, J. Kramer, R. Chatley, S. Uchitel, and H. Foster, "LTSA - Labelled Transition System Analyser," 2006.

33) F. Massacci, "Reasoning about Security: A Logic and a Decision Method for Role-Based Access Control," *International Joint Conference on Qualitative and Quantitative Practical Reasoning*, Vol. 1244, pp. 421–435, 1997.

34) W. McCune, "OTTER 3.3 Reference Manual and Guide," *Argonne National Laboratory, Illinois*, 2003.

35) Canegie Mellon, "The SMV System," 2003.

36) Microsoft, "Windows Server 2003 における Windows Rights Management Services,"

Microsoft TechNet, 2003.

37) K. Ong and R. M. Lee, "A Decision Support System for Bureaucratic Policy Administration: An Abductive Logic Programming Approach," *Decision Support Systems*, Vol. 16, No. 1, pp. 21–38, 1996.

38) Organization for The Advancement of Structured Information Standards (OASIS), "eXtensible Access Control Markup Language (XACML) Version 2.0," *OASIS Standard*, 2005.

39) J. Otten and W. Bibel, "leanCoP: Lean Connection-Based Theorem Proving," *Journal of Symbolic Computation*, Vol. 36, pp. 139–161, 2003.

40) C. Ribeiro, A. Zúquete, P. Ferreira, and P. Guedes, "Security Policy Consistency," *Technical Report, INESC*, 2000.

41) H. Sakurada, "Model Checking Configurations of Local Area Networks," 第 2 回ディペンダブルソフトウェアワークショップ論文集, 2005.

42) R. S. Sandhu, "The Typed Access Matrix Model," *IEEE Symposium on Security and Privacy*, pp. 122–136, 1992.

43) M. E. Stickel, "A Prolog Technology Theorem Prover: A New Exposition and Implementation in Prolog," *Technical Note 464, Artificial Intelligence Center*, 1989.

44) M. Strembeck, "Conflict Checking of Separation of Duty Constraints in RBAC - Implementation Experiences," *Conference on Software Engineering*, pp. 224–229, 2004.

45) G. Sutcliffe and C. Suttner, "The TPTP Problem Library: CNF Release v1.2.1," *Journal of Automated Reasoning*, Vol. 21, No. 2, pp. 177–203, 1998.

46) Adobe Systems, "Adobe LiveCycle Rights Management ES," データシート, 2008.

47) H.B. Wang, S. Jha, P. McDaniel, and M. Livny, "Security Policy Reconciliation in Distributed Computing Environments," *International Workshop on Policies for Distributed Systems and Networks*, pp. 137–146, 2004.

48) R. S. Weinstein, M. R. Descour, C. Liang, A. K. Bhattacharyya, A. R. Graham, J. R. Davis, K. M. Scott, L. Richter, E. A. Krupinski, J. Szymus, K. Kayser, and B. E. Dunn, "Telepathology Overview: From Concept to Implementation," *Hum Pathol*, Vol. 32, No. 12, pp. 1283–1299, 2001.

49) 河野和宏, 伊藤義道, 青山明史, 鴨田浩明, 馬場口登, "隣接行列を用いたアクセス制御ポリシーの統合法," 情報処理学会研究報告, Vol. 2007, No. 48, pp. 45–50, 2007.

50) 鴨田浩明, 河野和宏, 伊藤義道, 馬場口登, “モデル検査ツールによるポリシー整合性検証,” 情報処理学会研究報告, Vol. 2007, No. 16, pp. 441–446, 2007.
51) 鴨田浩明, 星川知之, 山岡正輝, 山本修一郎, “オンデマンド VPN システムの実装と評価,” 情報処理学会論文誌, Vol. 47, No. 8, 2006.
52) 経済産業省, “情報システム・ソフトウェアの信頼性及びセキュリティの取組強化に向けて-中間報告書,” 高度情報化社会における情報システム・ソフトウェアの信頼性及びセキュリティに関する研究会, 2009.
53) 高橋成文, 東川淳紀, 山本修一郎, 小尾高史, 谷内田益善, 大山永昭, “2 階層 PKI を用いたオンデマンド VPN システム,” 情報処理学会論文誌, Vol. 46, No. 5, pp. 1128–1136, 2005.
54) 次世代 IC カードシステム研究会, “NICSS 要件書 V1.20,” *NICSS-F WG*, 2004.
55) 社団法人日本情報システム・ユーザー協会, “障害を発生させない，被害を拡大させないためのシステム対策ガイド,” 重要インフラ情報システムの高信頼性対策調査プロジェクト, 2009.
56) 松尾尚文, パッタラリーラープルット, 土屋達弘, 菊野亨, “ホームネットワークシステムにおける連携サービスのモデル検査による検証,” 電子情報通信学会技術研究報告, Vol. 105, No. 597, pp. 7–12, 2006.
57) 早川晃弘, 星川知之, 高橋成文, 鎌仲裕久, “オンデマンド VPN における構成情報生成に関する一検討,” 情報処理学会第 67 回全国大会講演論文集, Vol. 3, pp. 329–330, 2005.
58) 竹内陽一, 早川晃弘, 高橋成文, 鎌仲裕久, “オンデマンド VPN におけるポリシー制御機能に関する一検討,” 情報処理学会第 67 回全国大会講演論文集, Vol. 3, pp. 331–332, 2005.
59) 独立行政法人情報処理推進機構, “重要インフラ情報システム信頼性研究会報告書,” 重要インフラ情報システム信頼性研究会, 2009.
60) 馬場達也, “マスタリング IPsec,” オライリー・ジャパン, 2006.
61) 馬場達也, 角将高, 藤本浩, 稲田勉, “動的 VLAN 制御による統合ワーム対策システムの提案,” 情報処理学会論文誌, Vol. 47, No. 8, pp. 2499–2511, 2006.
62) 有馬一閣, 鴨田浩明, 星川知之, 山岡正輝, “仮想アドレスを用いたプライベートネットワーク間相互通信方式のオンデマンド VPN への適用,” 情報処理学会研究報告, Vol. 2006, No. 26, pp. 103–108, 2006.

63) 有馬一閣, 鴨田浩明, 早川晃弘, 山岡正輝, “仮想アドレスを用いたプライベートネットワーク間相互通信方式,” 電子情報通信学会技術研究報告, Vol. 105, No. 208, pp. 7–12, 2005.

本論文に関する原著論文

[A] 論文

A1) H. Kamoda, M. Yamaoka, S. Matsuda, K. Broda, and M. Sloman, "Access Control Policy Analysis Using Free Variable Tableaux," *Transactions of Information Processing Society of Japan (IPSJ)*, Vol. 47, No. 5, pp. 1515–1529, 2006.

A2) 鴨田浩明, 星川知之, 山岡正輝, 山本修一郎, "オンデマンド VPN システムの実装と評価," 情報処理学会論文誌, Vol. 47, No. 8, 2006.

[B] 国際会議

B1) P. H. Deussen, I. Schieferdecker, and H. Kamoda, "A Methodology for Policy Conflict Detection Using Model Checking Techniques," *The 24th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE2004)*, pp. VII(1)–(14), 2004.

B2) H. Kamoda, M. Yamaoka, S. Matsuda, K. Broda, and M.Sloman, "Policy Conflict Analysis Using Free Variable Tableaux for Access Control in Web Services Environments," *The 14th International World Wide Web Conference (PM4W2005)*, pp. 5–12, 2005.

B3) H. Kamoda, A. Hayakawa, M. Yamaoka, S. Matsuda, K. Broda, and M. Sloman, "Policy Conflict Analysis Using Tableaux for On Demand VPN Framework," *The 6th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM2005)*, pp. 565–569, 2005.

B4) H. Kamoda, "Access Control Policy Verification Using Model-Checking," *The 1st Joint Workshop on Security Software Engineering in Tokyo*, p. 12, 2007.

B5) N. Ichihara, H. Kamoda, and H. Oguro, "Smartcard Security Development Using Formal Method Tool SPIN," *The 9th International Common Criteria Conference (ICCC2008)*, pp. 2405(1)–(15), 2008.

[C] 口頭発表

C1) 有馬一閣, 鴨田浩明, 早川晃弘, 山岡正輝, “仮想アドレスを用いたプライベートネットワーク間相互通信方式,” 電子情報通信学会技術研究報告, Vol. 105, No. 208, pp. 7–12, 2005.

C2) 有馬一閣, 鴨田浩明, 星川知之, 山岡正輝, “仮想アドレスを用いたプライベートネットワーク間相互通信方式のオンデマンド VPN への適用,” 情報処理学会研究報告, Vol. 2006, No. 26, pp. 103–108, 2006.

C3) 鴨田浩明, 河野和宏, 伊藤義道, 馬場口登, “モデル検査ツールによるポリシー整合性検証,” 情報処理学会研究報告, Vol. 2007, No. 16, pp. 441–446, 2007.