社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－DPS－126（52）
2006－CSEC－32（52）
2006／3／17

# 状況依存型サービスのための分散演繹機構の提案

大谷 隆三[†]　竹内 亨[†]　吉田 幹[*]　寺西 裕一[‡]　春本 要[◇]　下條 真司[‡]

[†] 大阪大学大学院情報科学研究科
[*] 株式会社ビービーアール
[◇] 大阪大学大学院工学研究科　[‡] 大阪大学サイバーメディアセンター

## 概要

ユビキタス環境では，状況に応じて提供するサービスの内容を切り替える状況依存型サービスの実現が期待される．状況を導き出すための手法としては，ルールを用いて情報間の関係を柔軟に導きだす演繹処理が有効である．しかし，既存の演繹処理を行うシステムでは情報を集約する必要があり，スケーラビリティの確保や情報のリアルタイム性，情報に対する柔軟な制御といったユビキタス環境において必要とされる要件を満たすことができない．そこで，本稿では分散環境で情報を集約することなく演繹処理を実行する分散演繹機構を提案する．また，分散演繹処理を行う上で発生する問題を解決するため，ルールの正規化手法とルール形式に応じた分散演繹処理のための問い合わせ手法を提案し，さらに，提案する問い合わせ手法の有効性を示すために，現実的な数値をパラメータとして設定した例を用いた評価を行った．

# Proposal of a Distributed Deduction Mechanism for Context-Aware Services

**Ryuzo Otani[†], Susumu Takeuchi[†], Mikio Yoshida [*], Yuuichi Teranishi[‡], Kaname Harumoto[◇] and Shinji Shimojo[‡]**

**[†] Graduate School of Information Science and Technology, Osaka University**
**[*] BBR Inc.**
**[◇] Graduate School of Engineering, Osaka University**
**[‡] Cyber Media Center, Osaka University**

## Abstract

In the ubiquitous environment, it is expected to realize context-aware services which change their contents according to user context. To derive user context, deduction reasoning is effective since it can derive the relationship between information dynamically and flexibly using the rules. Existing deduction reasoning systems assume the information needed for processing are collected beforehand, which causes the problems to realize scalability, freshness, and flexibility that are required in ubiquitous environment. In this paper, we propose an distributed deduction mechanism which can process the rules without collecting the required information for processing. We also propose a query algorithm based on the rule normalization method. We evaluated the effectiveness of the proposals using the realistic value.

## 1　はじめに

近年のユビキタス環境に対する関心の高まりとともに，いつでもどこでもユーザが自由に情報を発信・取得可能とする技術に関する研究開発が盛んに行われるようになってきている．

ユビキタス環境では，サーバやデスクトップ PC などの固定端末だけではなく，携帯電話や PDA，ラップトップ PC などのモバイル端末，街中に設置されたセンサ，家庭内の家電製品などの様々な機器がネットワークに接続し，それぞれの端末や機器（以降，ノードと呼ぶ）が情報を保持・発信することが想定される．

ユビキタス環境における典型的なアプリケーションとして，ユーザの状況に応じて，適切なコンテンツに切り替えて提示する状況依存型サービスが挙げられる．状況依存型サービスの例としては，ユーザが家を出かける際に今から行く予定の場所で雨が降っていれば傘を持って行くよう促すガイダンスを提示するサービス，昼食時間になるとユーザの移動手段が徒歩であれば 1km 以内，車であれば 5km 以内の飲食店リストが自動的に提示されるサービスなどが挙げられる．

状況依存型サービスにおいては，まずアプリケーションがどのようなコンテンツを提示すべきか判定できるように，ノードに保持されている情報をもとにしてユーザが置かれる状況を導き出す必要がある．状況を導き出すための手法としては，プログラムや設定ファイルにあらかじめ集めるべき情報およびそれらの組み合わせ方や判別方法などのロジックを埋め込んで指定することが考えられる．しかし，サービス提供に必要となる情報，及び，それらを処理するロジックをあらかじめプログラムに記述する場合，要求される情報や組み合わせ方が変化するたびにプログラムの内容を書き換える必要があり，アプリケーション開発者にとって大変な労力となる．

こうした問題を解決し，ユーザの状況を動的かつ柔軟に導き出すための手法として，演繹処理を用いることが考えられる．演繹処理では，あらかじめ記述されたルールと与えられた情報をもとに新たな情報を論理的に導き出す．ルールを利用して状況を論理的に導き出すことによって，動的に変化する情報間の関連付けが可能となる．また，同じ「近い」というルールに対して，移動手段が徒歩であれば 1km 以内，車であれば 5km 以内の距離というように状況に応じてルールを選択させることができるため，必要な情報を柔軟に組み合わせて利用することが可能となる．

ネットワーク上に存在する情報間の関係を記述し，演繹処理に利用するための枠組みとして Semantic Web [1] が標準化されつつある．これまでにいくつかの Semantic Web の実装が行なわれてきているが，いずれも，ルールや情報が集約されている環境を前提としたものであり，ユビキタス環境に十分対応できるものではなかった．

そこで本稿では，ユビキタス環境においてユーザの状況を導き出すための分散演繹機構を提案する．提案機構では，分散して存在する端末上でルールを用いた演繹処理を実行することで，リアルタイム性，スケーラビリティ，情報管理の柔軟性を確保した状況依存型サービスを実現させることが可能である．以下 2 章では，提案する分散演繹機構に対する要件，および分散化において考慮すべき実現上の問題点について述べる．3 章では，2 章で挙げた問題点を解決するためのルールの問い合わせ処理アルゴリズムについて述べ，4 章において現実的な数字を用いた評価を示す．

## 2 状況依存型サービスのための演繹機構

本章では，提案する分散演繹機構に求められる要件，および分散化において考慮すべき実現上の問題点について述べる．

### 2.1 演繹機構に対する要求

ユビキタス環境における状況依存型サービスを想定した演繹機構には以下が要求される．

- スケーラビリティの確保
  ユビキタス環境においては，いつでもどこでも，あらゆるユーザがネットワークに接続可能な状況が想定され，ユーザ数は膨大なものとなる．さらに，街中のいたるところに設置された温度センサ，位置センサなどがネットワークに接続され，それらから得られる情報を活用することを考えると，演繹機構においては膨大な量の情報源が分散して存在することを想定する必要がある．したがって，それらを現実的に処理可能なスケーラビリティの確保が必要である．

- リアルタイムな情報の反映
  状況依存型サービスを実現する上では，ユーザが置かれている現実の状況に即した情報の提示が要求される．例えば，1 日おきに収集された情報に基づく状況の判断では，時々刻々と変化するユーザの状況を反映しきれない．センサなどから得られる情報をリアルタイムに反映した演繹処理が要求される．

- 情報のアクセス制御
  ユーザの状況は，ユーザの位置情報やプロファイル情報などに基づいて導き出されるため，ユーザのプライバシに深く関わるものとなる．このような情報は，ユーザが好きなときに，誰に対して情報を開示するか否かを柔軟に制御できることが望ましい．

### 2.2 分散環境における演繹機構に対する要求

ネットワーク上に存在する情報間の関係を記述し，演繹処理に利用するための枠組みとして Semantic Web が標準化されつつある．これまでにいくつかの Semantic Web の実装が行なわれてきており，分散環境における演繹処理を実現するための機構として FaCT [2]，Racer [3]，DRAGO [4] などが提案されている．しかし，いずれも情報およびルールは，ネットワーク経由でひとつの端末上に集約された上で演繹処理が行われることが前提となっている．このように状況を導き出すために必要な情報がひとつの端末に収集される前提では，前節で示した要求，すなわち，スケーラビリティの確保，リアルタイムな情報の反映，情報のアクセス制御を実現することは困難である．

これらの要求を満たしつつ，分散環境で演繹処理を行うためには，ネットワーク上の各ノードに対して処理すべきゴールを送信し，各ノードにおい

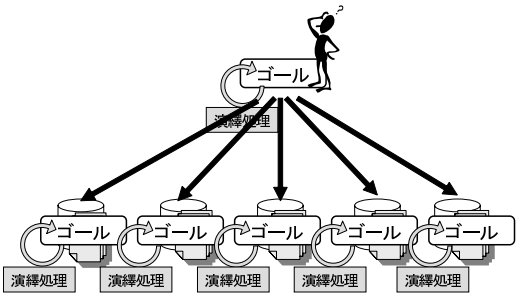


図 1: 分散環境における演繹機構の概要

て演繹処理を実行する必要がある (図 1)．各ノードで演繹処理を実行することで，サービスを享受するユーザのノードが情報を保持する端末から情報を集約する必要がないことから，スケーラビリティの確保，およびリアルタイムな情報の反映が可能であり，また，各端末が自ら情報を保持するため，いつでもアクセス制御する相手の変更などが可能であり，柔軟な管理が行える．もし各ノードにおいてさらにゴールを評価する必要が生じた場合は再帰的に処理を行う．

### 2.3 分散演繹機構を実現する際の問題

例えば，状況情報として「ユーザ otani が今日行く場所の天候」を得るためのゴールが以下の通り定義されたとする．

```
:-  weather(A, B), near(A, C),
    scheduledLocation(otani, C).
```

*weather* とは第 1 引数にあたる場所の天候を得る述語，*near* は第 1 引数の近くにあるオブジェクト (モノ，人など) を得る述語，*scheduledLocation* とは第 1 引数にあたるユーザの次のスケジュールが実施される場所を得るための述語を示している．ここで，それぞれのアトムを受け取った端末がどのような情報を返却するかについて考えてみる．

最初に記述されているアトム *weather(A, B)* を受け取った端末は，A および B の値を固定しないまま自らが得ることができる気象情報を全て返却することとなる．この場合，明らかに最終結果を導き出すために対象端末が返却する全ての情報が利用されることはなく，大量の無駄な結果送信が起きることとなる．

次のアトム *near(A, C)* を評価する際も同様の問題が生じるが，このアトムの場合すべての位置センサから周辺情報にあるオブジェクトの情報が返却されることとなる．位置センサも対象オブジェクトとして扱うこととすると，位置センサがカバーする範囲に重複する部分があった場合，重複した結果が返却されてしまう．

最後のアトム *scheduledLocation(otani, C)* は，ユーザ otani が持つスケジュール情報から場所に関する情報を得るものとなり，このアトムを満たす情報数は限定される．

先に評価した 2 つのアトムと最後に評価したアトムの違いは，引数に含まれる変数の数である．引数に含まれる変数が 2 つのアトムを送信すれば，返却される結果に無駄や重複が膨大に生じる可能性があり，分散環境においてはこのような無駄や重複はネットワークリソースの無駄使いとなるため，極力避けなければならない．よって，分散環境において演繹処理を行う上では，アトムの引数に含まれる変数の数を考慮して問い合わせを行う必要がある．

これまでに分散して存在する情報に対して演繹処理を行う試みとして，文献 [6] などが存在するが，上記のような問題について考慮されていなかった．

## 3 ルールの正規化と問い合わせ手法

本章では，2.3 節で説明した問題を解決するために，まず分散環境における問い合わせを考慮したルールの正規化手法を提案する．さらに，正規化したルールに基づいてあらかじめ問い合わせ手法を決定することで，分散環境における問い合わせの効率を向上する問い合わせアルゴリズムを提案する．

### 3.1 分散演繹機構におけるルール記述

本稿ではルール記述を一般的な論理プログラミングの手法に従い，ホーン節 (Horn clause) として定義する．ただし，RDF [5] における SPO (subject, predicate，object) の関係を扱うため，すべてのアトムは 2 引数の述語により定義されるものとする．また，RDF において predicate を変数として定義することが可能であるが，これを許すと高階述語論理の計算の枠組みが必要となるため，predicate は定数であると限定する，なお，ルールの構文には Prolog を用いる．

### 3.2 ルール形式の分類と正規化

#### 3.2.1 ルール形式の分類

ひとつのルールは左辺（ヘッド）に記述されるアトム（ヘッドアトム）の引数である 2 変数と，右辺（ボディ）を構成するアトム（ボディアトム）の引数との間に存在する関係から，以下の 3 種類の最も単純な構造により形成されるといえる．

**直列型ルール**

ヘッドアトムの引数である 2 変数の関係が，ボディアトムの引数の関係を遷移的に辿ることで導き出されるルールである．直列型ルールを Prolog の構文で記述すると以下のような式となる．

$p$(A, Z) :- $p_1$(A, B), $p_2$(B, C), ... , $p_n$(Y, Z).

**並列型ルール**

ヘッドアトムの引数である 2 変数が，全ボディアトムの引数である 2 変数と一致するルールである．

つまり，ヘッドアトムの引数として記述された 2 つの情報間に存在する複数の関係によって導き出されるルールである．並列型ルールを Prolog の構文で記述すると以下のような式となる．

$p$(A, Z) :- $p_1$(A, Z), $p_2$(A, Z), ... , $p_n$(A, Z).

**関係を持たないルール**

ヘッドアトムの引数である 2 変数が異なるボディアトムの変数として存在するが，並列型ルールのように 2 変数と同じボディアトムに含まれる変数が保持する関係を辿っても，ヘッドアトムの述語の引数である 2 変数の関係が導き出せないルールである．このルールを Prolog の構文で記述すると以下のような式となる．

$p$(A, Z) :- $p_1$(A, B), $p_2$(Y, Z).

いま説明したルール形式のうち，関係を持たないルールについては提案機構の目的である「情報間の関係を利用して情報検索を行う」という要求を満たさないため，本稿で取り扱うルールからは除外する．よって，本稿で利用する最も単純な形式のルールは直列型ルールと並列型ルールとなる．

### 3.2.2 ルールの正規化

すべてのルールは，述語に対して逆写像（インバース）と結合（コンビネーション）の演算を導入することによって，直列型と並列型の 2 種類のルール形式の組み合わせに正規化できる．

例えば，

```
:-  scheduledLocation(A, B),
    nearestStation(B, C),
    near(C, D),
    visitedRestaurant(A, D),
    recommendMenu(D, Z).
```

なるゴールは以下のような 2 つの直列型ルールと 1 つの並列型ルールに正規化できる．

```
                :-  comb1(A, D),
                    recommendMenu(D, Z).
comb1(A, D) :-  comb2(A, D),
                    visitedRestaurant(A, D).
comb2(A, D) :-  scheduledLocation(A, B),
                    nearestStation(B, C),
                    near(C, D).
```

## 3.3 分散演繹処理のための問い合わせ手法

本稿で想定する分散環境においては，特定のゴールを評価するために必要な情報を持つノードが特定できないため，ネットワーク上の対象となるノードに対してルールを送信することで問い合わせを行う必要がある．その際の問い合わせ手法は，以下の 2 種類に分類できると考えられる．

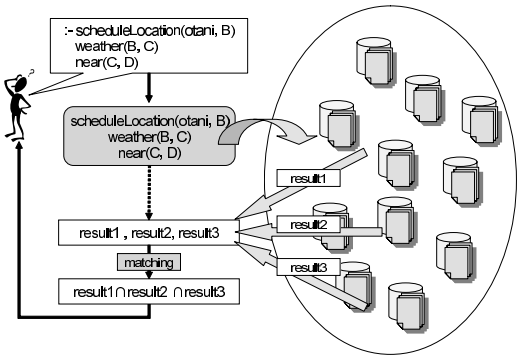


図 2: 同時検索手法

### 3.3.1 同時検索手法

ゴールを構成する複数のアトムを同時に送信し，ゴールを受信した端末は 1 つずつアトムを処理する．処理後，ゴールを満たす情報を問い合わせ送信元の端末に返信する．問い合わせ元の端末は，マッチングによって返信された結果の中からすべてのルールを満たす情報を探し出し，演繹処理の結果を導き出す．これを同時検索手法と呼ぶ（図 2）．

本手法は，問い合わせを行う端末の負荷が少なく，また問い合わせに該当する情報量が少ない場合，全体として素早く結果が得られることが期待できる．しかし，1 つのゴールに対して該当する情報の数が膨大となる場合，問い合わせ元の端末に対して膨大な数の情報が返信される可能性があり，問い合わせ内容によっては非現実的な手法であるといえる．

### 3.3.2 反復検索手法

アトムを 1 つずつ送信し，受信した端末がアトムを処理することで結果を送信元に返信する．問い合わせの送信元では，返信された情報をまだ送信されていないアトムに対して代入し，続けて問い合わせを行うことによって既知の結果を反映させたアトムを他の端末に送信する．これを反復検索手法と呼ぶ（図 3）．これによって条件を絞り込んだ問い合わせが可能になるため，同時検索手法のように該当する情報量が膨大になる場合でも，必要最小限の情報を送受することで処理が可能になる．しかし，アトムの数に比例して検索処理のステップ数が増加するという問題があるため，問い合わせ処理の平均的な所要時間が同時検索手法と比較して大きい手法であるといえる．

## 3.4 問い合わせアルゴリズム

3.3 節で述べたように，分散環境における問い合わせ処理は問い合わせ内容に応じて使い分ける必要がある．すなわち，問い合わせに該当する情報

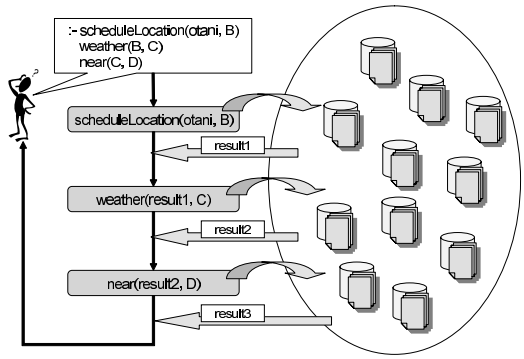


図 3: 反復検索手法

量が少ない場合はゴールを同時検索手法によって素早く求める必要があり，情報量が非常に多い場合は反復検索手法によって絞り込んだ問い合わせを繰り返す必要があると考えられる．したがって，3.2.1 節で述べた直列型ルールのように問い合わせが連続的に関係性を持つときは，それぞれの問い合わせに該当する情報量が極めて膨大となる可能性があることから，反復検索手法による問い合わせを行う必要があるといえる．また一方で，並列型ルールの場合，該当する情報量が多い場合を除き，それぞれの問い合わせは独立しているため，同時検索手法によって結果を求める時間を省くことが可能であると考えられる．

以上より，分散環境における演繹処理を，以下のような流れで行う．まず，他の端末に対して問い合わせを行うかを判断し，他の端末に問い合わせを行う必要があるときにルールの形式を判断する．

ルールの形式が直列型である場合は反復検索手法を適用する．このとき，引数が定数のアトムがあればそれを起点として反復検索を開始する．並列型である場合には同時検索手法を適用し，もし情報が大量に返信されてきた場合には，反復検索手法に切り替える．図 4 は，以上をフローチャートとして表現したものである．

## 4 問い合わせアルゴリズムの評価

3.4 節で提案した問い合わせアルゴリズムの有効性を確認するため，具体的なルールに対して現実的な数値を当てはめて評価を行った．比較対象としては，アトムを 1 つずつ問い合わせとして送信する最も単純な手法を想定した．

**直列型ルール**

先に示した,「ユーザ otani が今日行く場所の天候」を取得するためのゴールは直列型ルールである．ここでは直列型ルールとしてこのゴールを利用する．値が確定しているアトムを先頭としたゴールは以下の通りとなる．

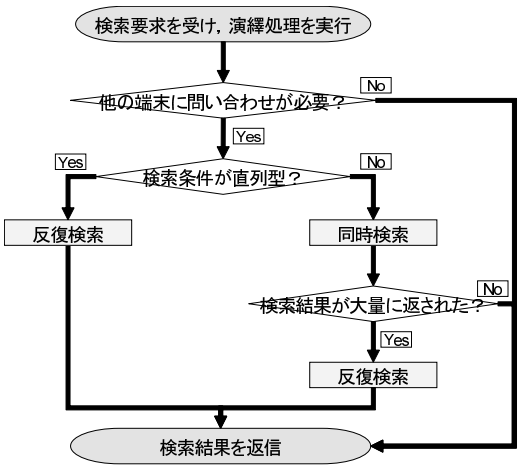


図 4: 分散演繹処理の流れ

```
:-  scheduledLocation(otani, B),
    near(B, C),
    weather(C, D).
```

このゴールに対して，パラメータとして表 1 に示す数値を設定し，評価を行った．気象センサは各市町村に 1 つ存在することを仮定した．したがって，日本国内の市町村数が約 3,000 であるため，日本国レベルでは，約 3,000 のセンサ数となる．また，位置センサとして，PHS 基地局による位置検出もしくは同などのものを想定し，一市町村に約 1,000，全国で約 16 万のセンサが存在すると仮定とした．また，全世界レベルでユビキタスネットワークが整備された場合の想定として，日本国レベルの約 400 倍の値（陸地面積/日本国の面積）を仮定している．ユーザ otani が持つ今日のスケジュールの数は 3 とした．

表 1: 直列型ルールの評価に利用したパラメータ

| | 市町村レベル | 国レベル | 世界レベル |
|---|---|---|---|
| 気象センサ | 1 | 3,000 | 1200,000 |
| 位置センサ | 1,000 | 160,000 | 64,000,000 |
| スケジュール | 3 | 3 | 3 |

提案問い合わせアルゴリズムに基づくこの例の問い合わせ結果数とステップ数は，表 2 で示される.「Naive Algorithm」は最も単純な手法を適用した結果であり,「Proposed Algorithm」が提案アルゴリズムを適用した結果である．単純な手法では，問い合わせ結果数が対象とするレベルに応じて膨大なものとなるのに対し，提案アルゴリズムでは，問い合わせ結果数はいずれの場合も 9 となり，提案アルゴリズムを利用することで問い合わせ結果として得られる数を大幅に削減することができるこ

表 2: 直列型ルールにおける問い合わせ手法の比較

| | 問い合わせ結果数 | | | ステップ数 |
|---|---|---|---|---|
| | 市町村 | 国 | 世界 | |
| Naive Algorithm | 1,000 | 16 万 | 6,500 万 | 3 |
| Proposed Algorithm | 9 | 9 | 9 | 3 |

とが分かる．ステップ数は，いずれの場合もルール数と同じ 3 である．

**並列型ルール**

並列型ルールの例として,「近くにあって行ったことがある好みの店」を取得するルールを利用する．このルールに基づいたゴールは以下の通り表現される．

```
:-  near(otani, B),
    visited(otani, B),
    favorite(otani, B).
```

このゴールに対して，パラメータとして「近くにある店」を 100 軒,「行ったことがある店」を 40 件,「好みの店」を 50 件，それぞれの共通部分を 4 件ずつある場合を仮定した．

単純な手法および提案アルゴリズムに基づくこの例の問い合わせ結果数とステップ数は，表 3 で示される．提案アルゴリズムを利用することで問い合わせ結果数とステップ数の両方を削減できることが分かる．

表 3: 並列型ルールにおける問い合わせ手法の比較

| 問い合わせ手法 | 問い合わせ結果数 | ステップ数 |
|---|---|---|
| Naive Algorithm | 190 | 3 |
| Proposal Algorithm | 174 | 1 |

## 5 考察

4 節の評価結果は，直列型，並列型それぞれの典型的な例による評価を示している．実際にはより複雑なルールから構成される問い合わせも考えられるが，3.2 節で示した通り，一般にルールは直列型，並列型それぞれの組み合わせと考えることができるため，複雑なルールであっても問い合わせ結果数やステップ数の削減は可能と考えられる．

また，前章の評価では問い合わせを全ての端末へ送信することを仮定しているため，ネットワーク上の問い合わせ送信のためのトラヒックの評価は行っていない．実際には，位置情報の問い合わせであれば，LL-Net [7] などの位置に基づく効率的検索のための P2P 技術を適用することによって，問い合わせトラヒックは削減することができると考えられる．問い合わせ順序を考慮する際には，このような技術の活用を検討する必要があるであろう．

## 6 まとめ

本稿では，ユビキタス環境における状況依存型サービスを実現するために必要となる状況を導き出すための情報を取得するための機構である分散演繹機構を提案した．提案機構では，ルールを他の端末に送信することによってスケーラビリティ・情報のリアルタイム性・情報に対するアクセス制御を実現している．

今後の課題としては，提案した機構の実装およびアプリケーションへの適用と問い合わせトラヒックの削減方法の考案が挙げられる．

## 謝辞

本研究の一部は，平成 15 年度総務省「ユビキタスネットワーク認証・エージェント技術の研究開発」の研究助成によるものである．

## 参考文献

[1] E. Miller, R. Swick, D. Brickley, B. McBride, J. Hendler, G. Schreiber, D. Wood, and D. Connolly, “Semantic Web.” available at http://www.w3.org/2001/sw/.

[2] I. Horrocks and P. F. Patel-Schneider, “FaCT and DLP,” in *Proceedings of the 7th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX ’98)*, pp. 27–30, May 1998.

[3] V. Haarslev and R. Möller, “RACER system description,” in *Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR ’01)*, pp. 701–706, June 2001.

[4] L. Serfini and A. Tamilin, “DRAGO: Distributed reasoning architecture for the semantic web,” in *Proceedings of the 2nd European Semantic Web Conference (ESWC2005)*, pp. 361–376, May 2005.

[5] G. Klyne and J. J. Carroll, “Resource Description Framework (RDF): Concepts and abstract syntax,” *W3C Recommendation*, Feb. 2004. available at http://www.w3.org/TR/rdf-concepts/.

[6] P. Adjiman, P. Chatalic, F. Goasdoué, and M.-C. Rousset, “Scalability study of peer-to-peer consequence finding,” in *Proceedings of 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pp. 351–356, July–Aug 2005.

[7] 金子 雄, 春本 要, 福村 真哉, 下條 真司, 西尾 章治郎, “ユビキタス環境における端末の位置情報に基づく P2P ネットワーク,” **情報処理学会論文誌: データベース**, vol. 46, pp. 1–15, Dec. 2005.