卒業論文 2007年度 (平成19年度)

曖昧さを考慮したコンテキスト解析手法の構築

指導教員

慶應義塾大学 環境情報学部

徳田 英幸

村井 純

楠本 博之

中村 修

高汐 一紀

重近 範行

湧川 隆次

Rodney Van Meter

慶應義塾大学 環境情報学部

藤本 俊佑

shunsuke@ht.sfc.keio.ac.jp

卒業論文要旨 2007年度 (平成19年度)

曖昧さを考慮したコンテキスト解析手法の構築

近年ユビキタスコンピューティングの発展に伴い、センサネットワーク技術も飛躍的に進歩している。私たちは従来よりも小型で高性能なセンサを、非常に安価な値段で入手可能となった。センサネットワークを利用することで、人が直接データを入力することなく、室内の状況や機器の動作状態などをコンピュータ自身が自律制御するシステムを構築できる。こうしたシステムが世間に広まることで、私たちを取り巻く環境はより便利になり、コンピュータと現実世界の距離が縮まるだろう。

一方、こうしたセンサネットワークから収集される温度や照度といった生データは、そのままの形式では利用しにくく応用範囲が限られてしまう。そのため膨大な量の生データを解析し、その中からモノの状態や人の動作といった意味ある高次情報を取り出すことが非常に重要である。このようにセンサデータから取り出される意味ある情報をコンテキスト、低次データから高次のコンテキストを取り出すことをコンテキスト解析と呼ぶ。コンテキスト解析を行うことで、収集されたセンサデータからアプリケーションが必要な情報だけを取り出すことができる。また、コンテキスト解析をアプリケーションと分離することによって、異なるセンサを用いたシステム同士の結合や、他のアプリケーションからコンテキストだけを利用することが可能となる。

本論文では、コンテキスト解析エンジンの柔軟性や再利用性といった視点から、従来手法では対応できなかった問題について考察する。その上で、問題を解決するために曖昧さを考慮したコンテキスト解析手法 Ambiguity-based Context Analysis System (ACAS) の提案を行う。ACAS では曖昧さの表現にファジィ理論というソフトコンピューティングの概念を利用する。コンテキスト解析に曖昧さという要素を導入することで、従来手法では対応が困難であった環境の動的な変化に対応させることを目標とする。本論文では ACAS を評価するためのプロトタイプを実装し、従来手法の一つであるルールベース手法との比較を行った。その結果、ACAS を用いることでより柔軟で再利用性の高いコンテキスト解析が可能となることを証明した。

キーワード：
ユビキタスコンピューティング, センサネットワーク, コンテキスト解析
コンテキストアウェア, ルールベース, 曖昧さ, ファジィ理論

慶應義塾大学 環境情報学部
藤本 俊佑

Abstract of Bachelor's Thesis

ACAS: Ambiguity-based Context Analysis System

Sensor network technology has seen rapid progress with the advance of ubiquitous computing. Cheap and smaller senor nodes are available on an annual basis, and we could easily construct a sensor network, in which sensor nodes would sense environmental information without human input. These systems would highly improvise our quality from life from an environmental perspective, and greatly cut down the distance between reality and virtual reality.

On the other hand, raw data from sensor network such as temperature data or illumination data are difficult to use. For that reason data analysis from raw data and extract advanced information from data is highly required. We define significant information taken out of the sensor data as context, and extract higher order of low level data as context analysis. Context analysis simplify extract necessary information from enormous amount of sensor data. Also context analysis can analyze from combination of different type sensors or application.

In this thesis, we evaluate problems of former context evaluation methods from a flexibility and reusability point of view, and introduce a context evaluation technique, ACAS to solve the problem mentioned above. ACAS uses a software architecture called Fuzzy Logic. Bringing a fuzzy evaluation technique into research would able former context evaluation adapt to difficult sensing environment under today 's research field. In this thesis, we implemented a prototype and compared it with the former rule-base method. As a result, we proved a more flexible and reusable context filtering method by using our ACAS method.

Keyword :

Ubiquitous Computing, Sensor Network, Context Analysis, Context-aware Rule-base, Ambiguity, Fuzzy Logic

Shunsuke Fujimoto

Faculty of Environment and Information Studies
Keio University

目 次

図 目 次

表 目 次

第1章　序論

本章では、研究の背景を述べた後、本論文の目的を提示する。そして論文全体の構成についてまとめる。

1.1　本研究の背景

計算機技術の発達により、巨大で、重量があり、高価であるという従来のコンピュータのイメージとは異なる、超小型コンピュータが実用可能な時代になってきている。こうした超小型コンピュータはオフィスや研究室にある高性能なデスクトップPC、ビジネス用途に用いられる携帯性に優れたモバイルPCのようなものではなく、小型プロセッサと限られたメモリ、センサデバイスと単一の無線通信モジュールなどによって構成される。超小型コンピュータは機能を制限して、ハードウェアの小型化を突き詰めることにより、数センチ四方の大きさにコンピュータとして最低限の処理能力と通信能力を持たせることに成功している。具体的な例としては、カリフォルニア大学バークレー校で開発されたMOTE [17] や東京大学森川研究室のU3 (U-cube) [18] などが挙げられる。これらの詳細については第2章で改めて解説する。

これらのコンピュータは非常に小さく値段も安いことから、これまで物理的にコンピュータを付けられなかったものにも取り付けることができるようになった。例えば、精密機器や小物類、食品の一つ一つに取り付ければ、現在の物流システムの効率化とより一層の透明化が可能となる。このように、超小型コンピュータを利用することで、物理的にネットワークに接続不可能な製品をより賢くすることができる。ただしこの賢さとは、モノ自体にコンピュータが内蔵されて、それ単体として新たな処理機能を持つことではない。モノの機能はそのままで、コンピュータに内蔵されたセンサによって空間の情報やサービスの内容、現在の状態を把握し、コンピュータ同士がその情報を共有することで協調動作を可能にするという意味である。コンピュータを小型化し、人から見えないところで動作させて知らず知らずのうちにその恩恵を受けられるようにすることを、Mark Weiserはコンピュータがあらゆるところに遍在するという意味のユビキタス [15] という言葉で表現した。現在は、多くの研究者が異種サービスを連動させて、人々の生活を支援するユビキタス社会の実現を目指して研究を行っている。

本論文ではこれ以降、小型コンピュータのことをセンサと言い換えて記述する。そ

の理由としては、この小型コンピュータのインプットはほぼ全てがセンサデバイスによるセンシングデータであり、ノードがそれ自体でPCのような高度な処理をすることはないからである。コンピュータというと多くの人が対話型のコンピュータを想像すると考えられるため、そのような誤解を与えないためにもセンサという言葉を用いる。

ユビキタス社会を実現するためには実空間の状態を把握した上で、コンピュータ自身が自律的に機能を選択してサービスを行うことが求められる。そのような仕組みを考えると、システムは実空間と仮想空間という2つの部分に分けることができる。実空間は私たちが生活している世界を表し、仮想空間はコンピュータ内部で処理される情報を意味する。実空間における人の行動や空間の状態などは、センサデバイスやタグによってデータとして収集され、コンテキスト解析によって、システムで利用可能な意味ある情報へと抽象化が行われる。本論文では、データを解析して得られる人の行動や環境の変化などの情報をコンテキストと呼ぶ。コンテキストを利用することで、仮想空間においても実空間の状況の把握が可能となる。そのような意味からコンテキスト解析は、実空間と仮想空間とを介するインタフェースであると考えられる。図1.1にインタフェースとしてのコンテキスト解析の概要図を載せた。図に示したとおり、コンテキストとして人の行動と空間の状態が分かるだけでもさまざまなサービスを構築することができる。

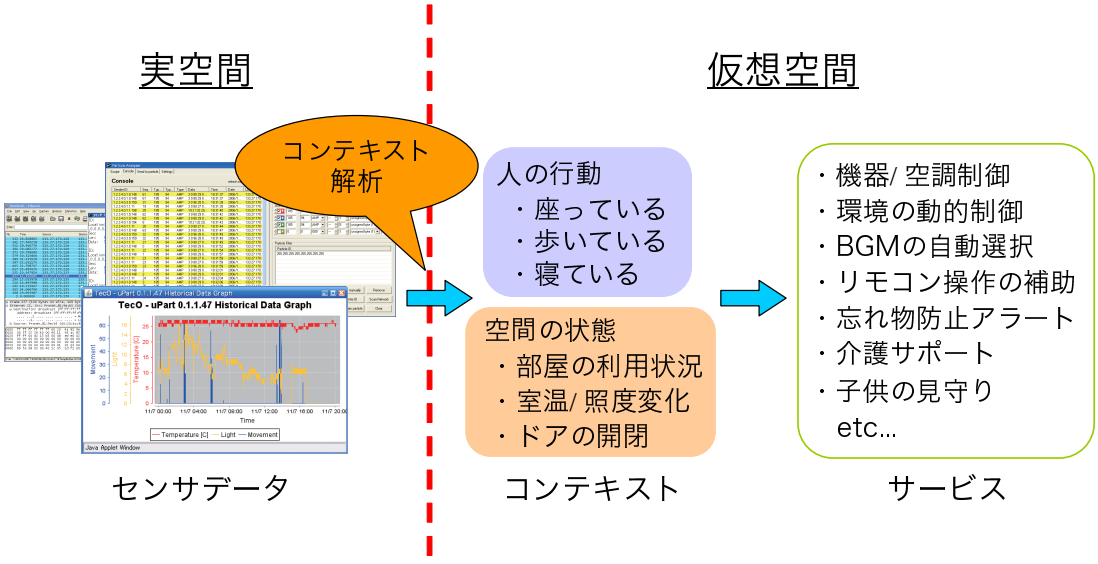


図1.1: インタフェースとしてのコンテキスト解析

実空間から仮想空間へとデータを受け渡す際にコンテキスト化する理由としては、コンテキストとして一度抽象化を行うことによって、センサデータの数値そのものを意識する必要がなくなり、開発者はセンサデータの値を直接見なくてもアプリケーションの開発が可能になるからである。例えば人の行動のコンテキストとしては「立つ」

「歩く」「座る」などが考えられるが、それを「センサ A が 0.4、センサ B が 0.7 なのでこの状態は座っている状態である」などと全てのデータと状態とを直接結びつけるようなことをするのは非常に効率が悪く、かつ汎用性に乏しいシステムとなってしまう。これをコンテキストとして抽象化することでシステム内で共通のコンテキストが利用できるため開発コストも抑えることができる。

コンテキストをもとにして、何らかの動作を行い実空間に影響を与えるものをアクチュエータと言う。例えば空調制御コンテキストアウェアシステムでは、室内の快適さをコンテキスト、快適さによって制御されるエアコンがアクチュエータとなる。コンテキストアウェアシステムではコンテキストを求めるだけではなく、実世界にその解析結果を反映させるためのデバイスに対する操作も考慮する必要がある。

センサデータからコンテキストを取得して、その状況に応じた機器を動作させることでさまざまなユビキタスサービスが実現できる。そのようなサービスの代表的なものとしては、環境の動的制御 [23] やユーザの動作補助 [24]、介護サポートや子供の見守りシステム [20] などがある。このようにコンテキストを介して、人間がシステムに対して直接の操作や命令をすることなく、環境からサービスを働きかけるシステムのことをコンテキストアウェアシステムと呼ぶ。

1.2 本研究の目的

本論文では、コンテキストアウェアシステムを実現するためのコンテキスト解析手法である Ambiguity-based Context Analysis System (ACAS) を提案する。ACAS は「曖昧さ」をコンテキスト解析に利用する解析手法である。入力値のセンサデータに対して重み付けをした上でコンテキスト解析を行うため、従来の手法では考慮することができなかったコンテキストの度合いを出力することができる。ここで言う度合いとは、私たちが日常的に言語化している「少し」「たくさん」「強く」など程度に幅を持つ指標のことである。度合いによって、アクチュエータに必要な動作量の値が求められるため、より効率的できめ細やかな機器制御が可能となる。

従来の解析手法 [10] [25] では、If-Then 文によるルールベース手法により閾値を一定の値に決めた上でコンテキスト解析を行うため、環境によってルールや閾値を手動で変更する必要があった。また、コンテキストを定義するために条件を複雑に構成する必要があることも、システムを煩雑にする原因となっていた。そのため、これら既存手法は柔軟性と再利用性に欠けるコンテキスト解析であったと言える。本論文で提案する ACAS では、柔軟性と再利用性の向上を目的としている。ユーザのイレギュラーな行動や環境の動的な変化に対応し、アプリケーションに依存しないコンテキスト解析手法を提供する。開発者は、アプリケーション内でコンテキストを利用するだけで良いため、生産性が向上し、システム開発における煩雑さを解消できる。

1.3　本論文の構成

本論文は全７章から構成される、各章の詳細は以下の通りである。

第１章　序論

本章では本論文の背景として、コンテキストアウェアシステムの概要と本研究の目的について述べた。

第２章　研究背景

コンテキストアウェアシステム全体の構成要素について述べ、実際に運用されているシステムから、現状での問題点を洗い出しそれらをまとめる。

第３章　研究の方針と概要

本論文で仮定する環境と想定シナリオを述べ、従来のコンテキスト解析における問題点を明らかにする。その解決手法として ACAS の構想を述べる。

第４章　曖昧さの表現方法

ファジィ理論の詳細を説明する。ファジィ理論によって曖昧さをどのように定義するのかを考察し、コンテキストとの関連性について述べる。

第５章　ACAS の設計と実装

ACAS の設計と実装について述べる。

第６章　ACAS の評価

ACAS と従来のルールベースによるコンテキスト解析手法との比較を行い、ACAS のコンテキスト解析精度や柔軟性などについて評価を行う。

第７章　結論

今後の課題と展望を述べ、本論文をまとめる。

第2章　研究背景

本章では、研究背景としてコンテキストアウェアシステムの特徴を述べる。そして、現状のコンテキストアウェアシステムの課題を考察し、コンテキスト解析の重要性についてまとめる。

2.1　コンテキストアウェアシステムとは

まず本論文における、コンテキストアウェアシステムの定義を述べる。コンテキストを直訳すると「文脈・前後関係・状況」などと表現されるが、ここでは人やモノ、空間の情報を収集しその中の関連性によって導き出す一つの状態のことを指す。コンテキスト解析とは、センサデータや位置情報などの多面的な情報から物事の関係性を読み取り、ある特徴を持った状況へと分類することである。そして、コンテキストアウェアシステムとはコンテキストを実際にシステムの動作制御に組み込んだものであり、センサデータから直接動作制御しているような既存のアプリケーションとは異なる。

表 2.1: コンテキストアウェアシステムの例

コンテキストアウェアシステム	センサアプリケーション
室内環境の統合的制御	自動ドア
老人・子供の見守り	火災報知器
忘れ物ブザー	既存のエアコン
物流システムの商品管理	自動照明装置
情報への透過的なアクセス	エスカレータの自動操作

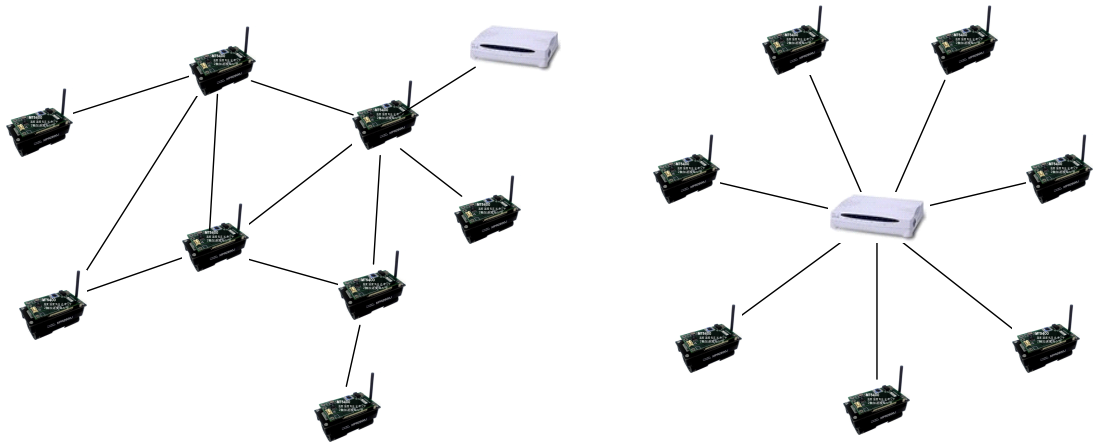
広義では、センサアプリケーションもコンテキストアウェアシステムであると考えられるが、本論文では明示的に別物として扱う。表 2.1 のようなセンサアプリケーションは、センシングと実際の機器の動作とが密接に関係しているため、センサだけ、機器だけを別のものに取り替えることはできない。それに対して、コンテキストアウェアシステムはコンテキストの情報を正確に得られさえすれば、センサはどのハードウェアでも良く、デバイスの取替えが可能である。また、センサデータのみを別のアプリケーションから利用することでシステムを協調的に動作させることができる。

2.2　コンテキストアウェアシステムの要素

本節ではコンテキストアウェアシステムを構成する要素について、センサネットワーク、センサデバイス、アクチュエータの 3 点から説明する。

2.2.1　センサネットワーク

データの集約方法の違いによって、センサネットワークはメッシュ型とスター型という 2 つの形態に分けられる。図 2.1 に概略図を載せる。センサデバイスがマルチホップ通信可能であればメッシュ型とスター型の両方を構築できるが、シングルホップ通信のみの場合は、スター型のセンサネットワークしか構築できない。メッシュ型とスター型のセンサネットワークはそれぞれ利点と欠点があり、以下にその特徴をまとめる。



メッシュ型センサネットワーク　　スター型センサネットワーク

図 2.1: メッシュ型とスター型

- **メッシュ型センサネットワーク**

  メッシュ型センサネットワークは、一般的なネットワーク通信技術の一つであるアドホックネットワークに近く、ノード同士でデータのやり取りを行うことでネットワークを形成する。ノードはセンサデータの送信フェーズに移ると近隣のノードを見つけ出し、次々とノードを中継することでデータの集約を行う。メッシュ型センサネットワークは対障害性に優れており、ネットワークを構成する中のいくつかのノードが利用不可能になった場合でも、その時点で通信可能な相手を見つけ出してデータの送信経路を動的に変更することで障害を回避する。複数の送信経路を持てる反面、データの衝突や経路制御アルゴリズムなどを考える必要があるため、構築に手間がかかるなどの問題を持つ。

- **スター型センサネットワーク**

  スター型センサネットワークは、全てのセンサデータを一つのノードに集中して送信することでデータの集約を行う。データをまとめて受け取るノードのことをシンクノードと言い、シンクノードは受け取ったデータをサーバ PC やローカルエリアネットワークに送信する。シンクノード以外の他のノード同士で通信をすることはないため、シンクノードに障害が発生するとその周辺のノードからのデータを受信できなくなる、単一故障点による問題が発生する。また、一つのノードが通信可能な範囲でしかデータを収集できないため、広範囲をスター型ネットワークで覆うためにはシンクノードを複数用意してエリアごとにセンシングする必要がある。

2.2.2 センサデバイス

次に、センサネットワークで用いられる主要なセンサについて、その性能を詳しく見る。また、本論文の実験で用いたセンサの詳細も述べる。

MOTE

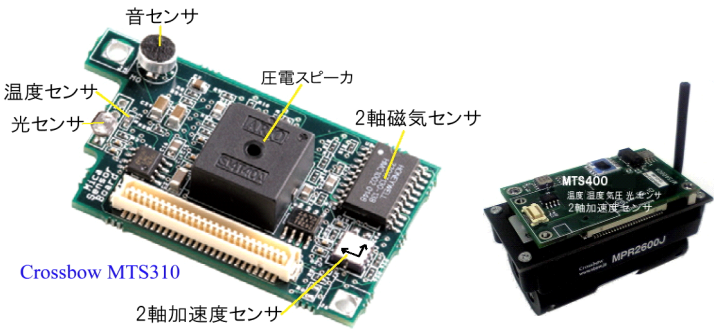


図 2.2: 無線センサデバイス MOTE

MOTE はカリフォルニア大学バークレー校で開発され、現在は Crossbow Technology, Inc. [1] から販売されている無線センサデバイスである。図 2.2 に MOTE の写真を載せる。MOTE は NesC 言語で実装された TinyOS [6] という独自のオペレーティングシステムによって動作する。TinyOS は MOTE 上で動作させることを目的としているので、CPU やメモリなど限られた資源を有効に使うための最適化がなされている。

MOTE はマルチホップ通信が可能であり、データの送受信を行う基板をベースに、必要なセンサをその上にモジュールとして取り付けることができる。データの送受信

を行う基板は MICAz と呼ばれ、周波数帯は 2.4GHz で IEEE 802.15.4 の無線規格を使用している。無線通信の基板とセンサの基板とを分けることによって、デバイスをモジュール化することを実現している。MICAz 上に乗せるセンサ付き基板の一つに MTS310 があり、図 2.2 の左側に拡大写真を載せた。このセンサで取得できるデータは音・光・温度・2 軸加速度・磁気センサ値である。センサ基板は自由に取り替えることができるため、用途に応じて使い分けることができる。また、ノードの上で集約されたセンサデータを PC で取得するには、専用のインタフェース基板が必要となる。この基板に MICAz を一つ取り付けると、そのノードがシンクノードの振る舞いをするようになり、これによってデータの送受信が可能になる。

MOTE には TinyOS 上で動作する擬似データベースとして、TinyDB [7] [8] というシステムが提案されている。TinyDB は MOTE で構成されるセンサネットワークを擬似的にデータベースと見立てて、そこに SQL を独自拡張したクエリを発行する。これによりデータベースで SQL を利用するのと同様の感覚で、センサネットワークからデータを収集できる。クエリを投げると、条件にマッチしたデータのみクエリに適合した形式で送信するので、無駄な通信の削減とバッテリーの省電力化が可能になる。また、各ノードが自立的に処理を受け持つので、障害が発生した場合でも部分的に正確なデータを収集することができる。

U3 (U-cube)

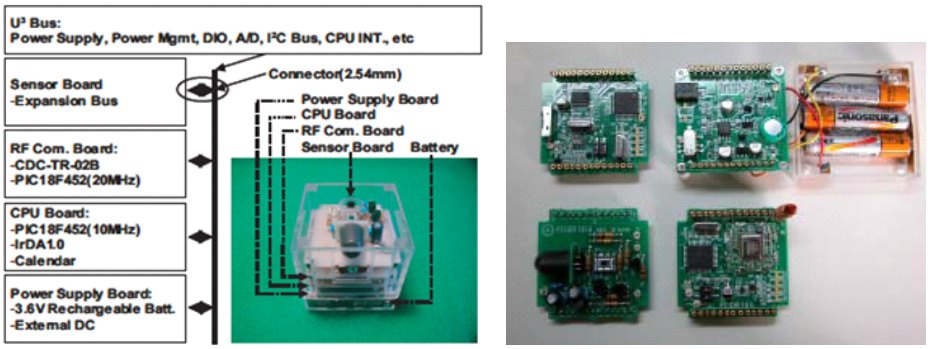


図 2.3: 無線センサデバイス U-cube

U-cube [18] は、東京大学森川研究室で開発されたキューブ型の無線センサデバイスである。図 2.3 に U-cube の写真を載せる。プラスチックキューブの大きさは 50mm × 50mm × 50mm で、その中に電源ボード・CPU・無線デバイス・センサボードの 4 つの基板が内包される。それぞれ独立した基板に分けることで、ハードウェアのモジュール化をはかっている。各基板は MOTE と同規格のバスコネクタで接続されているた

め、MOTE とセンサボードの互換も視野に入れた設計がなされている。315MHz 帯無線と IrDA の赤外線インタフェースの 2 つの通信規格を備え、IrDA を利用することでユーザプログラムのダイナミックロードが可能である。センサボードでは照度・温度・人体感知センサの値が取得でき、このボードも用途に応じて取り替えることができる。

U-cube ではハードウェアを制御するための API の仕様を定義してあるので、物理層と上位層の分離が可能であり、この点で MOTE との差別化をはかっている。通信プロトコルだけでなく送受信を行うデータフォーマットも独自の形式を定義している。そのため従来のセンサデバイスよりも、データサイズの縮小やエネルギーの効率化を可能にしている。彼らは U-cube の人体感知センサを利用して、室内の人の動きを検知するアプリケーションを構築し、評価を取っている。その結果、パケット送信のリアルタイム性と通信相手選択の公平性にまだ問題はあるが、これらはアプリケーションに依存するものであり各モジュールごとにプログラムの機能を入れ替えることで解決できるので、U-cube の開発モジュールとしての有用性を証明している。

uPart

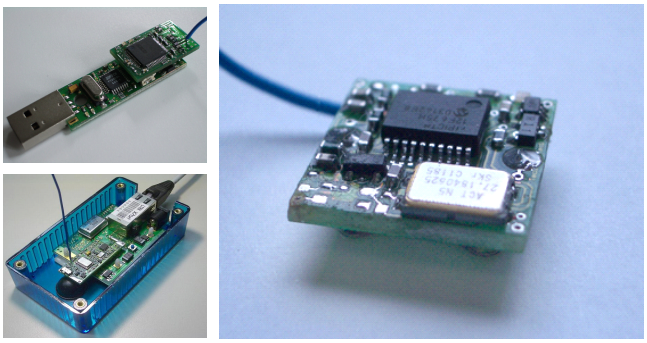


図 2.4: 無線センサデバイス uPart

uPart はドイツのカールスルーエ大学 TecO 研究所 [13] で開発された小型無線センサデバイスである。図 2.4 に uPart の写真を載せる。uPart の大きさは 1.5cm 四方ほどで、取得できるセンサデータの値は温度・照度・振動である。振動数はボールスイッチによって計測されるため、加速度センさのように振動の向きや強弱などは正確に測れない。シングルホップセンサであるため、データの受信にはシンクノードが必要となる。全てのセンサデータはシンクノードが受信し、そこからサーバ PC やネットワークに向けて送信される。図 2.4 の左側に写るデバイスが uPart のシンクノードで、上を USB Bridge 下を XBridge と言う。uPart はスター型センサネットワークを構築するため、広範囲のセンサ情報を収集する場合は、エリアごとにシンクノードを複数設置してデータの受信を行う必要がある。

uPart にはCPUやメモリなどの資源は、必要最低限のものしか内蔵されていない。メモリが非常に少ないのでセンサ上でユーザプログラムを処理することはできないが、その分センサの大きさを小さくすることとバッテリーの省電力化に成功している。電源はボタン電池であり、30秒毎の送信頻度ならば数ヶ月程度バッテリーが持続する。uPart は小型かつ安価であるため、センシング対象を選ばず、日用品などにも手軽に取り付けることができる。

センサデバイスのまとめ

以上3つのセンサデバイスについて特徴を見てきたが、表 2.2 にそれぞれのハードウェア性能をまとめた。この表から、センサの大きさが小さくなればなるほど機能が限定されることが分かる。センサ上で分散処理をするには、マルチホップ通信が可能であり、ある程度のメモリ領域が確保されていることが必須である。この条件を満たすセンサデバイスは、これらの中では MOTE と U-cube である。uPart ではユーザプログラムを動作させるだけのメモリ容量が確保できない。以上のことから、センサを協調させて処理を行うことと、センサの小型化はトレードオフであることが分かる。

また、本論文のシステム評価には uPart を使用している。これは徳田・高汐研究室において、uPart を室内に配置してモノや空間のデータをデータベースに収集するプロジェクトが行われているためである。実際に uPart を利用してコンテキスト解析を行うと、本節でまとめた点に加えて uPart の実機特有の問題が発生する。最も厄介なのは、振動センサの動作が非常に不安定な点である。システム構築では実機を運用して初めて見えてくる問題も多く、本説での問題点の列挙は代表的なものに留めてある。このような実装上の問題点は、第5章 ACAS の設計と実装 の章にて詳しく解説する。

表 2.2: センサデバイスのハードウェア性能

項目	MOTE	U-cube	uPart
CPU	7.37 MHz	10 MHz	4 MHz
メモリ	128 KByte	32 KByte	1.4 KByte
通信周波数帯	2.4 GHz	315 MHz	315 MHz
マルチホップ	可能	可能	不可
電源	単三乾電池	充電電池	ボタン電池
センサデータ	音/温度/照度/ 2軸加速度/磁気	温度/照度/人体感知	照度/温度/振動
センサの価格	3〜6万円	数万円以上	3000円程度
センサの大きさ	65 × 35 × 40 mm	50 × 50 × 50 mm	15 × 15 × 7 mm

2.2.3　アクチュエータ

次にコンテキスト解析結果から、実空間へ影響を及ぼすデバイスであるアクチュエータについて述べる。現在のコンテキストアウェアシステムにおいて、アクチュエータと考えられるものは大きく 2 つに分類される。一つは環境に内包されて動作するもの、もう一つは持ち運べる大きさの小型デバイスによって動作するものである。

環境に内包されているアクチュエータ

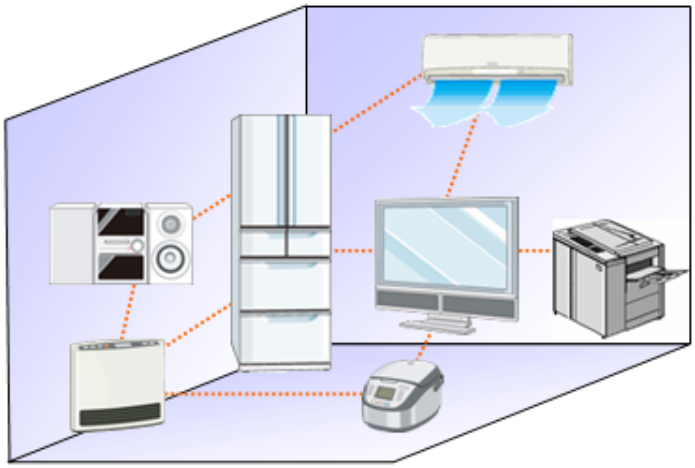


図 2.5: 環境に内包されているアクチュエータの例

環境に内包されているアクチュエータとして代表的ものは、図 2.5 で示した生活家電である。それ以外にも、オフィスや商業施設などに配備されている大規模な空調管理システムや、公共空間で不特定多数の人が利用する施設案内標示なども環境に内包するアクチュエータと考えられる。これらの機器を組み込むことで、人が直接機器操作を行ったり動作を把握しなくても、環境が自ら判断して人に合わせた動作を行うようなユビキタス空間を実現できる。

従来の機器制御は、対象となるパラメータを平均化した上で隔たり無く動作させることを目標としていた。例えば、空調システムでは室内の温度を一元的な数値で表現し、エアコンの温度設定を行う。しかし、コンテキストによる状況判断を行えば、個人に焦点を当てた機器制御が可能となる。これは個人の嗜好をコンテキストの一つに定義し、システムが動作の関連性などを学習することで可能となる。空調制御を例に挙げると、室温設定は常温になっている環境で、室内に暑がりの人がいる場合は、普通に冷風を送るのではなく若干強めに風を吹かせ、逆に寒がりの人がいるような場合は少し温風を吹かせるような動作が考えられる。コンテキストを利用するとこのようなパーソナライズされた動作が可能になる。

小型デバイス型アクチュエータ

図 2.6: 小型デバイス型アクチュエータの例

図 2.6 に示したとおり、小型デバイス型のアクチュエータの代表的なものとしては、携帯電話や PDA、ポータブルゲームなど片手に収まる大きさの電子機器がある。最近ではアクチュエータを身に付けるという発想から、ウェアラブルコンピュータとしての衣類や腕時計、メガネなどもアクチュエータと考えられる。これらはデバイスが動作して人や空間に影響を与えるというより、情報の出力先としての機能が求められる。

小型デバイスは特定の人が常に携帯しているという前提の下、さまざまなコンテキストアウェアシステムに利用される。システムの動作イメージとしては、自分がどこにいて、何をやろうとしているのかを環境にばらまかれたセンサによってセンシングする。その情報を元にコンテキストが解析されて、小型デバイス上に情報の表示や、利用者の周辺にある機器の制御などのサポートを実現する。

例えば携帯電話の位置情報取得システムを利用して、周囲の店の情報を集めるシステムを考えた場合、これまではどの店は何がおいしくて、値段はどれくらいなのかという情報を逐一調べてから、ユーザ自身が比較することで自分の好みに適した店を決めていた。これをコンテキストウェアシステムで実現すると、ユーザの好みといったメタ情報は既に端末に入力されていて常に携帯している状態にあると考えられ。イタリアンの店で、混雑していなくて、値段もお手ごろで、目的地への通り道にある、などという非常に細かい条件の判断も、コンテキストを複数掛け合わせた複合的な判断によって判断することができるため、ユーザ自身が何か特別な操作を行う必要がなくなる。コンピュータにあまり馴染みのないユーザでも、難しい操作を必要としないのでより簡単にすばやく目的の情報にたどり着くことができるようになる。

2.3 コンテキストアウェアシステムフレームワーク

本節ではコンテキストアウェアシステムを構築するためのフレームワークについて説明する。一般的な例として、ウェブアプリケーションとの連携を行うフレームワークと、データプロセッシングを利用したコンテキストの定義に関するフレームワークの 2 つを取り上げる。

2.3.1 ウェブアプリケーションとの連携

コンテキストアウェアシステムとウェブは密接な関係にあり、ウェブインタフェースとの協調は必要不可欠である。Michael Hinz らは論文 [3] にて、ウェブアプリケーション上でコンテキストを容易に扱えるようにするコンポーネントベースのフレームワークを提案している。Michael らは、ウェブ上でさまざまなコンテキスト情報を扱うためには、sensing, processing, representing に対してそれぞれ統一された形式を与える必要があり、そのためにはシステムを 3 つの層に分けた上で、コンポーネントとしてセンサデバイスやコンテキストの定義を扱う必要があると提言している。

各層の役割を簡単に述べると、最下層は Sensor Layer と呼ばれ、センサデバイスごとに得られるデータの形式や種類が異なるという差異を吸収する。この層では、データ取得のための共通のインタフェースを与え、それを HTTP Request にて送信する。HTTP Request には CC/PP という W3C によって策定された共通フォーマットを利用している。CC/PP はユーザの嗜好や機器の特性を記述するためのもので、デバイスに依存しない記述方法を可能にするものである。2 層目の Context Modeling Layer では、Sensor Layer で抽象化されたセンサデータを各モデルによって解析する。Michael らはここに外部のコンテキスト定義モデルを利用することを提案している。3 層目の Context Model Layer では、2 層目で判別したコンテキスト情報から Device Profile や Location Profile などの属性情報に変換したものを出力する。出力形式は複数あり、標準的なものは XML による出力となる。XML 形式に変換して出力することで、ウェブアプリケーション上でコンテキスト情報を容易に利用できる。彼らはこのフレームワークを用いてプロトタイプの実装を行っており、デバイスの処理能力に応じてウェブの表示内容を変化させるデモを構築した。

2.3.2 データストリームプロセッシング

センサデータからコンテキストを生成する手法の一つとして、Markus C. Huebscher らは論文 [5] にて、コンテキストアウェアシステムにおけるコンテキストの信頼性を

常に高い状態に保つことを可能にする、データストリームプロセッシングを応用したミドルウェアを提案している。

このミドルウェアでは、初めに最低限のコンテキストに関する定義を与えるだけで、その後はコンテキストを動的に生成することができる。システムの流れは、センサネットワークからの生データを受信すると、それをコンテキストプロバイダーと呼ばれるコンテキストを生成するためのメタデータの集合に変換する。コンテキストプロバイダーはデータベースに対して、既にそのコンテキスト情報が定義済みのものであるかを検索する。そしてコンテキストサービスとして登録済みであれば、そのコンテキストに対応したアプリケーションを実行する。もし新たなサービスである場合は、データベースにアプリケーション情報との関連性を記述して、次に同様のコンテキストが現れたときにスムーズにサービスへ移行できるようにする。

このミドルウェアでは QoC (Quality of Context) という指標を定義している。コンテキストの精度や信頼性の低い場合は、そのコンテキストを供給しているコンテキストプロバイダーを切り替えることでより高品質なコンテキスト情報を利用できるとしている。コンテキストアウェアシステムの起動中に、センサの故障やネットワークの障害などで著しくコンテキストの信頼性が落ちた場合、同じコンテキストを定義している他のセンサデータによってコンテキスト解析を続けることで、品質を落とさずにサービスが持続可能となる。

2.4 統合的なコンテキストアウェアシステム

近年、コンテキストアウェアシステムを構築するための実験環境が配備されている。本節ではそのような実験環境のうち代表的なものとして、慶應義塾大学の Smart Space Laboratory、Smart Living Room、uPlatea、東京大学の STONE ルームというユビキタステストベットについて説明する。

Smart Space Laboratory (SSLab)

Smart Space Laboratory [11] は慶應義塾大学徳田・高汐研究室 [28] で開発されたユビキタス技術の実験環境である。SSLab の定義するユビキタス空間とは、ネットワークにつながれたコンピュータがユーザの周辺環境に配備され、それがコンピュータであると分からない形で遍在している環境のことを指す。オブジェクトとしては、大型ディスプレイとプロジェクタのほか、複数のセンサ、照明機器、A/V 機器などが配置される。各デバイスや機器はネットワーク接続することで、相互通信が可能となる。さまざまなプロトコルが混在した環境を擬似的に作り出すことで、ヘテロジニアスネットワークの実現と、仮想情報家電の運用実験を行うことができる。

図 2.7: Smart Space Laboratory

図 2.8: Smart Living Room

このほかリビングルームをスマートスペース環境の対象と捉えたSmart Living Roomという部屋も構築されている。こちらではテレビやソファ、テーブルなどが備え付けられて、人がリラックスしてくつろぐ環境を想定している。カメラによる画像解析技術を利用してユーザが誰なのかを認識し、対象ごとに振る舞いをかえるシステムや、アンビエントな環境情報通知システムなどが開発されている。

uPlatea

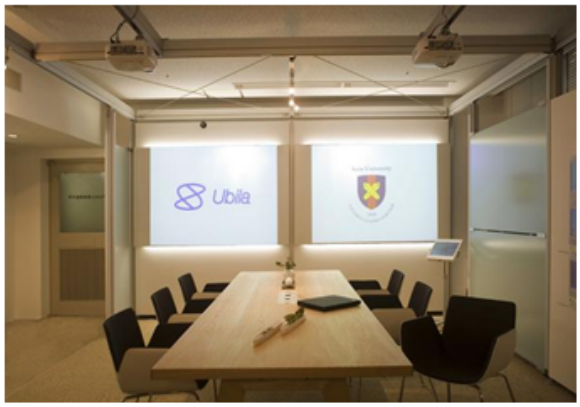


図 2.9: uPlatea

uPlatea [14] はSSLabを発展させたユビキタス実証空間である。エリアを3つに分けて、シチュエーションごとにユビキタス技術を提案している。それぞれのエリアは、ミーティングルーム、公共空間、リビングルームをイメージしており、ユーザが通常の生活行動を取る際に、どのようなサービスを実現できるのかデモで見せることができる。uPlateaは、ユビキタス技術を実際に体験することに重点が置かれており、ユビキタスの未来をよりリアルに感じることができる。

STONE ルーム

図 2.10: STONE ルーム

STONE [29] とは「Service synThesizer On the NEt」の大文字の部分を取った名前で、コンピュータがネットワークで相互接続される環境において、さまざまなサービスを統合するミドルウェアのことを指す。この研究の実験環境として、東京大学の青山・森川研究室 [26] では STONE ルームというテストベットを構築している。

STONE ルームには、ディスプレイ、カメラ、プリンタなど一般的なコンピュータ機器のほか、超音波受信機や赤外線等のデバイスが備え付けられ、互いにネットワークでつながっている。STONE ルームのアプリケーション例としては、超音波センサによってユーザの位置を特定し、システムがユーザに最も近い場所にある機器を動的に選択するシステムがある。これは従来のようにユーザが利用する機器を選ぶ必要がないため、より直感的な機器の利用をサポートすることができる。

STONE はデバイスに対してインプットとアウトプットの形式を記述することで、サービスを透過的に受けることを可能にする。例えば、印刷する際にプリンタの出力形式がその印刷物に対応していないというようなことがある。従来このような場合には、何らかのアプリケーションを使って印刷物のフォーマットを変更したのち、再び印刷する必要があった。しかし同様のシステムを STONE を利用して構築すると、インプットとアウトプットの情報は把握されているので、システムが出力形式の違いを検知する。そして、システムが動的にフォーマットの変更手続きを行い、ユーザは出力形式が異なることを知らなくても何事もなく印刷することができる。

これまで機器ごとに規格やインタフェースが異なることが、システムを柔軟に構築できない原因であった。STONE はデータ形式の違いを吸収し、ユーザがプロトコルの差異を意識することなく利用できるアプリケーションの構築を目標としている。

2.5　コンテキストアウェアシステムにおける問題

本節では、コンテキストアウェアシステムを運用する上での問題について述べる。それぞれハードウェアに起因する問題と、ソフトウェアに起因する問題の 2 つに分けて考察する。

2.5.1　ハードウェアの問題

ハードウェアが直接の問題となるのは、デバイスの物理的な設置方法や、デバイスで取得するセンサデータの形式に関する場合である。これらはコンテキストアウェアシステムを構築した後に問題が表面化することがある。それらの特徴を以下にまとめる。

- **デバイスの再設置**

  コンテキストアウェアシステムの初回構築時は問題にならないが、しばらく時間が経過してからセンサをより高機能なものに取り替える際に、センサの再設置法は問題となる。センサの再配置はもとより、ID の再マッピングには大幅な人的労力がかかる。また、環境に埋め込まれたセンサや、非常に大掛かりな組み込み機器の場合は、そのデバイスだけを取り替えるということは非常に難しい。

- **データ形式**

  本章の表 2.2 からも分かるとおり、センサにはさまざまな種類があり、センサごとにセンサデータの形式が異なる。温度や照度など一般的なデータでも、センサによってはデータフィールドの幅が異なっており、そのままでは正しくデータの比較ができない場合がある。また、センサに限らずデバイスのサポートしている規格に微妙な差異があることも問題である。ユビキタス環境では、ネットワークにつながる機器は全てが協調可能であることを目標としているため、特定のプロトコル依存によって利用できないような機器があってはならない。

- **デバイス間の通信**

  現在のコンテキストアウェアシステムにおいて、データはネットワークに送られた後、データベースに蓄積されるか、サーバ上で処理されてシステムに利用されるかどちらかの行程を進む。しかしこの構成では、ネットワークやサーバに障害が発生した場合、システムが利用できなくなるなど、深刻な状態に陥る可能性がある。それを回避するためには、ネットワークを経由しないで、その場にある機器同士が通信することが求められる。そのためデバイス間通信を可能にする、物理的なインタフェースが必要となる。

2.5.2 ソフトウェアの問題

ソフトウェアにおいては、各インタフェースをどのように設計するか、コンテキストアウェアシステムの初期設定・メンテナンス方法などが問題となる。本論文で論じる、コンテキスト解析における問題もここに含まれる。

- **インタフェースの設計**

  センサデータからコンテキスト解析、コンテキスト解析結果から実空間の状態の推測、実空間の状態からアクチュエータへの動作命令など、コンテキストアウェアシステムには、情報の抽象化を行う部分が複数ある。そのため、各インタフェースをどのように設計するのかは、システムの柔軟性や拡張性にかかわる重要な問題である。一般的には、ミドルウェアやフレームワークとしての解決法が提案される。

- **初期設定とメンテナンス**

  センサの値は利用環境によって取得値にばらつきがある。そのためコンテキストを正確に判別するためには、環境ごとの設定が必須である。その作業はシステムの初回構築時だけでなく、情報の更新作業やデバイスのメンテナンス時にも行わなければならない。また、ユーザが情報やセンサを後から追加することも考えられるので、一度構築して終わりというような実装ではなく、メンテナンス性も考慮に入れる必要がある。コンテキストアウェアシステムの導入にかかる手間としては、センサの設置に次いで重要な項目であるので、大規模なシステムであるほど慎重な設計を行い、メンテナンス性を保持しなければならない。

- **コンテキスト解析**

  コンテキスト解析で最も重要視されるのは、コンテキストの精度である。コンテキストアウェアシステムの実現のためには、常に正確なコンテキストの解析結果が求められる。例えば、センサの故障や外部的な要因から想定外の状況に陥った場合にも、最低限のコンテキストの信頼性を保たなければならない。そのためコンテキスト解析に関しては、解析の閾値をどのように決定するのか、環境が動的に変化しても正しい解析が可能か、コンテキストを組み合わせてより抽象度の高いコンテキストにすることなどに対処可能でなければならない。また、高機能なセンサを利用せず、一般的な機能のみのセンサを利用して、コンテキスト解析が可能となるように、デバイスに依存しない解析手法として実装する必要もある。このためには、センサデータだけの情報を解析するだけではなく、それ以外のさまざまな情報、例えばタグ情報やメタ情報、ウェブからのデータの読み出しなどを総合的に扱って、コンテキスト解析を行う必要がある。

2.6　本章のまとめ

本章では、コンテキストアウェアシステムを構成する要素、代表的なフレームワーク、統合的なコンテキストアウェアシステムのテストベットについて述べた。コンテキストアウェアシステムは、センサデータの受信、センサデータからのコンテキスト解析、解析されたコンテキストに対してアクチュエーションを行う、という3つのフェーズから成り立つことを述べた。また、現状のコンテキストアウェアシステムの問題を考察することで、システムの今後の課題をまとめた。そして、本論文で問題とするコンテキスト解析における、一般的な問題について論じた。以上で、本章の研究背景に関する記述を終える。次章からは、本論文で問題とするコンテキスト解析手法のより詳しい記述に入る。

第3章　研究の方針

本節では、本論文の研究方針を述べる。まず、従来のコンテキスト解析手法における問題点をまとめ、その解決方法として Ambiguity-based Context Analysis System (ACAS) の提案を行う。

3.1　研究概要

第 2 章でまとめた問題点の中で、コンテキスト解析に焦点を当てるということは既に述べた。本論文の最終的な目標は、従来の手法よりも柔軟性と再利用性の高いコンテキスト解析手法を構築することである。まず、問題点を明確にするため、従来のコンテキスト解析では不十分な点をまとめる。そして、それらを解決するために、曖昧さをコンテキスト解析に用いる利点を述べ、ACAS の機能要件を挙げる。

3.1.1　従来のコンテキスト解析の問題点

コンテキスト解析における問題点として、特に重要であると考えられる 5 つについて考察していく。ACAS の提案によってこれらを改善できるようにする。

1. **センサデータの誤差**
2. **センサデータの不到達性**
3. **コンテキストの表現方法**
4. **環境の動的な変化**
5. **アルゴリズムの移植性**

センサデータの誤差

センサは一つ一つの部品が小さく、数センチの基板の中にマイコンやセンサデバイスなどが密集したハードウェア構成になっている。そのため、各デバイスは精度よりも部品の大きさに重点を置いて選ばれている。コストを抑えた製品であるため、高性能なハードウェアと異なり、取得値に数ミリ単位の誤差が含まれるのは避けられない。したがって、誤差をあらかじめ計算に入れた上でコンテキスト解析を行う必要がある。

しかし、従来のコンテキスト解析手法では外れ値も同様に処理を行ってしまうため、正確な判別ができているとは言えない。即時的な動作を必要とするアプリケーションなど、より正確なコンテキスト解析が求められる場合に問題となる。

センサデータの不到達性

センサデータの不到達性もコンテキスト解析にとって深刻な問題となる。図 3.1 のグラフは、uPart の送信頻度とパケット衝突率の関係を表している。縦軸の値は正常にデータが送信された割合で、横軸はデータの送信頻度を示す。ノード数の増加に伴い、パケットの到達性が低くなることが分かる。このグラフから、ノード数が 100 を超えるような大規模なセンシングの場合、送信頻度が 10 秒では、その半数のパケットが衝突することが分かる。これでは、データが正常に受信できているかの確証が持てず、コンテキスト解析にも影響が出てしまう。

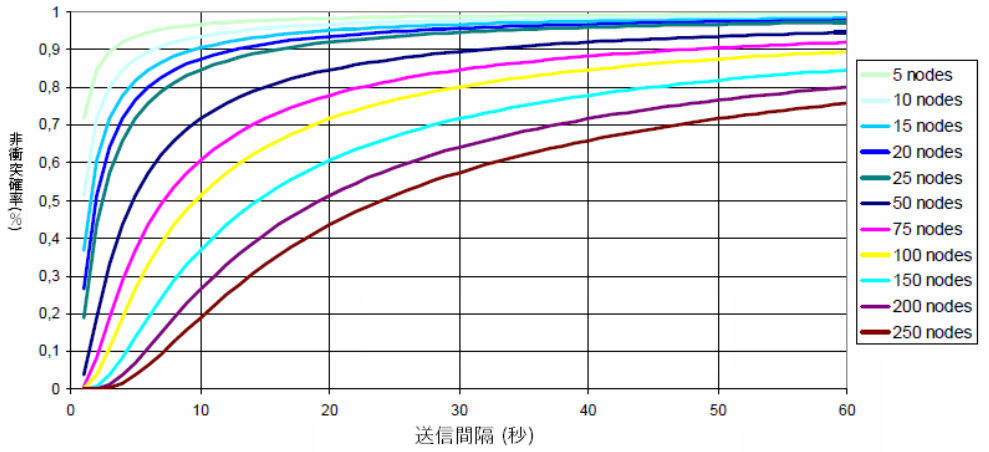


図 3.1: uPart におけるパケット衝突率

センサデータの不到達性は大きく 2 種類の状態に分けられる。

- データがほとんど受信されず、数時間から数日に 1 回データが得られる
- 常にデータが受信されおり、まれにデータが到達しなくなる

前者では、ほとんどデータは得られないということを解析に含んだ上で、センサデータ以外の情報もあわせてコンテキストの推測を行うことが重要である。一方、後者では常にデータを受け取れている状態が定常状態であると考えられるので、むしろデータが欠損したことは何らかの状況の変化が起こったと考えることもできる。つまり、センサと受信デバイスとの間に何らかの障害物が通ったために通信がさえぎられた場合などの状況が考えられる。そのため、センサデータの補完よりもなぜデータが得られなかったのかを、コンテキストの変化と取ることが必要となる。

コンテキストの表現方法

コンテキストの定義は、アプリケーションに依存することが多い。一つのアプリケーションにしか利用できない実装では、再利用性に乏しく、他のシステムとの連携が行えないなどの問題も発生する。そのため、アプリケーションに依存しない定義の方法として、さまざまな研究者がコンテキストを統一的に記述するためのフレームワークやミドルウェアを提唱している。その多くは、センサの状態の記述を組み合わせることにより、コンテキストを表現している。以下に具体的な事例を述べる。

西垣弘二らは論文 [25] にて、コンテキスト解析のためのルール記述言語の提案を行っている。コンテキスト解析結果からあるアクチュエータに動作の命令を出す際に、複数のコンテキストが同時発生し、一つのアクチュエータに対して別々の命令を出すような状況が起こりうる。このフレームワークは、より優先度の高いコンテキストからの命令を実行するためのルールの記述法を提案している。そして、コンテキストだけでなくアクチュエータに対しても、同様のルールを適応させることで情報家電の連携が可能になるとしている。一方、小林英嗣らは報告書 [22] で、コンテキストをルールベースの形でユーザが登録するのは、取得できるコンテキストの種類や数が多くなるにつれて困難になることが問題であると述べている。そのためルールを If-Then などと直接記述するのではなく、コンテキストのログとその状態を映したカメラの動画、そして各センサデータとを、後からマッチングすることでコンテキスト定義を行う手法を提案している。この手法はセンサデータと実際のコンテキストとを確実に結びつけることができる反面、その設定したセンサデータとほぼ同じ値にならなければコンテキストとして認識できない。そのためコンテキストの定義の後、推論エンジンを利用することでセンサデータにある程度の幅を持たせている。

環境の動的な変化

既存の解析手法では定常状態を定めた上で、センサデータの変化からコンテキストを判別している。センサデータの変化を判断するためのルールは、システム構築時に定数で定められることが多い。そのため環境が急激に変化した際には、正しくコンテキスト解析ができない。これは閾値が固定的であることに起因する問題である。

図 3.2 は夏と冬のセンサデータの取得値の変化を時間単位で見たものである。表 3.1 にデータの測定条件を載せた。センサはミーティングルームの机の側面に取り付けられたもので、それぞれ 10 日間の受信データの平均値をグラフに示した。これらは、季節は夏と冬で異なるが他は同条件であるにも関わらず、温度と照度の取れ方に差があることが分かる。このように季節が変化するだけで、同じ条件であっても取得データには大きな変化が現れる。つまり、センサデータは常に一定の波形のものが取れるとは限らないことが分かる。そのため閾値を固定的に定義すると、実際に環境が急激に

変化したときに正しいコンテキストが把握できず、システムに問題が起こることがある。より短時間での環境変化では、大きなばらつきが現れると考えられるため、コンテキスト解析において、環境の動的な変化に対応させることと条件が固定的にならないようにすることは重要な問題である。

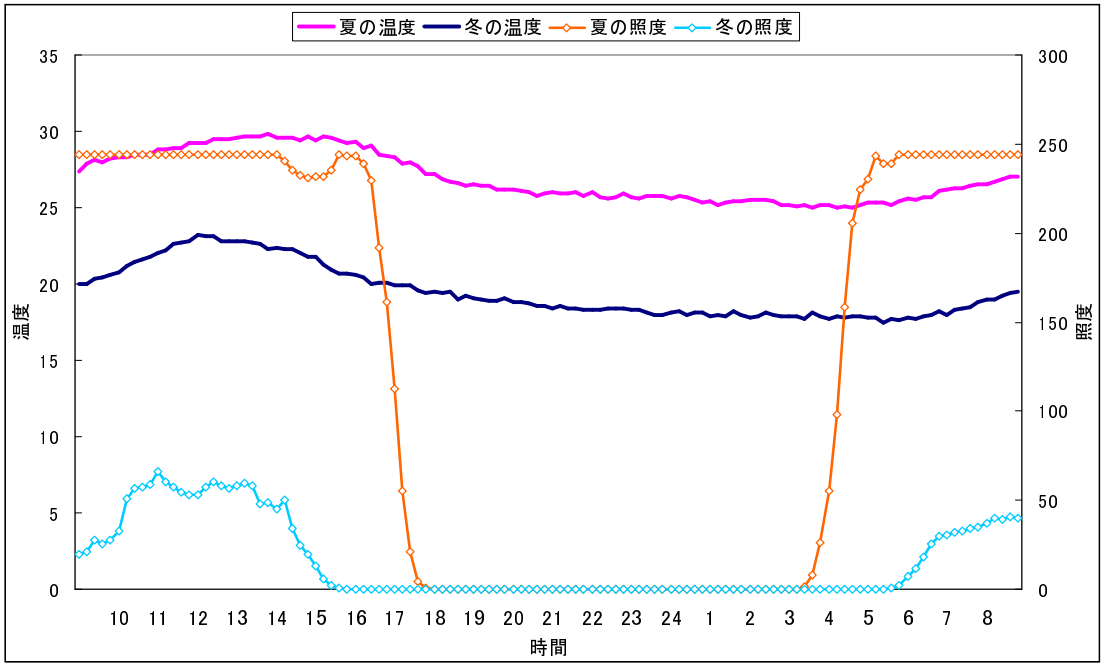


図 3.2: 夏と冬のセンサデータの変化

表 3.1: センサデータの測定条件

項目	内容
日時	夏 (8/10〜8/20), 冬 (12/20〜12/30)
センサ	uPart (送信間隔 10 秒)
受信データ	温度 / 照度
設置場所	SSLab の机の側面
データ数	約 8000/日

アルゴリズムの移植性

従来のコンテキスト解析は、特定のアプリケーションに依存した実装を行っていることが多い。つまり、ある環境下でのコンテキスト解析が可能であればよいという考えのもとでシステム全体が設計・構築されている。そのため、別の環境に新たにシステムを構築する場合、センサデバイスの変更などにより既存のコンテキスト解析アル

ゴリズムがそのままの状態で利用できないことがある。図 3.3 に概略図を示す。これについて以下のようなシステムを想定して説明する。

左側 (アプリケーション A)：ミーティングルームが使用中か調べる
右側 (アプリケーション B)：教室で講義が行われているか調べる

既に左側のシステムが構築されていたとき、それを右側のようなシステムに再利用することを考える。まず、センサデバイスの差異やセンサの設置場所、周辺環境の状態によって取得できる値が異なるなど物理的な問題が発生する。これによりアルゴリズム内で閾値の変更が必要になるため、そのような環境依存の部分に対してその差異を吸収するようなモジュールが必要となる。またアプリケーションだけを見た場合、どちらのシステムも複数の人間がその場にいるのかという情報を欲しているのにもかかわらず、コンテキストとしての定義が異なるので左側で定義したコンテキスト情報をそのまま再利用することは難しい。そのため、低次のコンテキストと高次のコンテキストというように、再利用を考えたコンテキスト定義をする必要がある。このように、アプリケーションが変わろうとも、最小限の手間でコンテキストのアルゴリズム部分を再利用できるようにしなければならない。

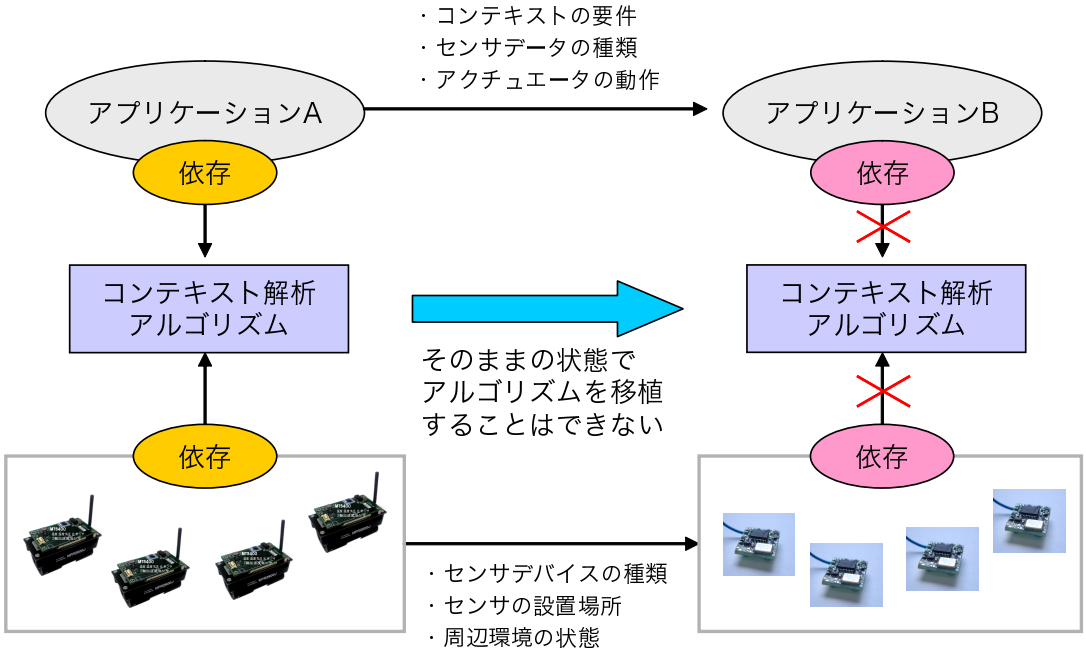


図 3.3: アルゴリズムを再利用できない場合

3.1.2　曖昧さを導入した解決策

本節では、3.1.1 節で述べた問題点に対して、曖昧さを導入することでどのような解決策が考えられるのかについて述べる。

センサデータの誤差

センサデータに誤差が生じるのは避けられないことである。一般的な解決策としては、コンテキスト判別の閾値を数%上下に広げることや、同じコンテキストを判断するメソッドを複数用意しておいて、双方の解析結果からより信頼できるものを選択することなどが考えられる。また実際には、センサデータの誤差とデータの変化をどのように区別するのかというのが非常に問題になることがあるが、本論文ではシナリオと環境をあらかじめ設定した上で実験を行うためここでは考慮しないことにする。

その上で、コンテキスト解析に曖昧さを利用すればセンサデータの誤差の問題を簡単に解決することができる。センサデータを曖昧さを含む集合によって表現し、その集合を用いたコンテキスト解析を行えば、誤差を曖昧さによって吸収することができる。そのため、開発者があらかじめ誤差を計算に入れてコンテキスト解析アルゴリズムを記述する必要はなく、その分の手間を削減することができる。

センサデータの不到達性

センサデータの不到達性は、システムとして必ず考慮に入れなければならない重要な問題である。前節で述べたとおり、不到達性には 2 種類の状況が考えられ、それぞれの対応策が求められる。これはコンテキストを利用するアプリケーションに依存する部分ではあるが、推論機構をコンテキスト解析に導入することは必須であり、それぞれの状況に合わせた最適なアルゴリズムを選択する必要がある。例えば、時間内に一定数以上のデータが到達しないと、正確なコンテキストが得られない場合は、常に推論エンジンを通した解析が必要となる。逆に、データが到達しなかったことからコンテキストを求める状況では、そのための推論方法が必要になる。

ただし、この問題も曖昧さによって簡単に対処することができる。曖昧さを利用すると、データの不到達という事象も曖昧さの表現により内包させることができる。そのため、通常通りコンテキスト解析を行えばデータの不到達性は考慮された上での解析結果が得られる。つまり、センサデータの誤差と同様に、曖昧さによる集合でセンサデータを定義すれば、曖昧さの中に不到達性が吸収されることになり、開発者が特に意識する必要はなくなる。

コンテキストの表現

コンテキストの表現方法に関しては、定義方法を統一的に扱うためのフレームワークを新たに構築するような方法は取らず、コンテキストの構造化を行うことで問題を解決できるよう試みる。コンテキストの構造化とは、コンテキスト A はコンテキスト B とコンテキスト C の集合というような、構造化表現を用いることで、それぞれの要素の分解と再構成を柔軟に行えるようにするものである。つまり、あるコンテキスト定義の中に、複数の基本的なコンテキストが内在して、それら全ての状況から上位のコンテキストを導き出すことである。そして、コンテキストの構造化をより効率的に行うために、コンテキストの定義は言語による、人間の論理思考の記述に似た形式で構築する。このような形式を選択した理由は、人間の論理思考と曖昧さとは非常に相性が良いからである。この定義方法によって、コンテキストの定義に際して開発者がより自然に理解できるようになると考えられる。

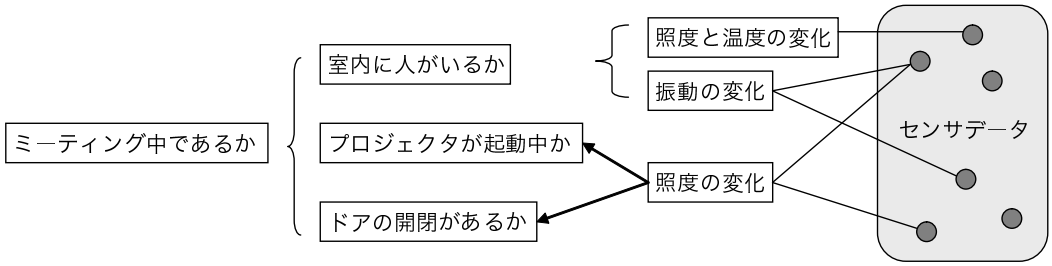


図 3.4: コンテキストの構造化

環境の動的な変化

環境の動的な変化に対応するためには、ある特定のセンサデータを基準センサとして閾値の幅を動的に変更できるような実装を行うか、環境の変化もコンテキストの一つとして定義してしまい、その上で他のコンテキストを導き出すことなどが考えられる。しかし、これもこの問題も曖昧さをコンテキスト解析に組み込むことで簡単に解決することができる。曖昧さとは境界があやふやなことを指すので、このような閾値が動的に変化するという状況は、むしろ常に想定していることである。したがって、曖昧さによって定義される範囲で、環境の動的な変化が起こった場合は十分に対応可能であると考えられる。

アルゴリズムの移植性

移植性の問題は、コンテキストを利用するアプリケーションに依存した、アルゴリズムのハードコーディングが原因となっている。そのため、コンテキスト解析アルゴリズムとアプリケーションにおけるコンテキストの利用形式を、それぞれ独立させて

記述する必要がある。アプリケーションの仕様やセンサ等のハードウェアに依存するプログラムはできるだけモジュール化するか、データの形式だけを定義して再利用できるようにする。解決策としては、プログラムはできる限りモジュール化して、特定の環境に依存しないアルゴリズムにすることを心がけなければならない。

曖昧さを導入した解決策のまとめ

以上のように、従来の問題点に対して曖昧さを導入した解決策について考察した。アルゴリズムの移植性とコンテキストの表現に関しては、実際にどのようなシステムを想定しているのかなどに強く影響される。一方、センサデータの誤差、センサデータの不到達性、環境の動的な変化に対しては、曖昧さによるセンサデータの表現とそれをコンテキスト解析に適用することで十分な効果が得られると考えている。

3.2 ACASの提案

本節では、曖昧さを導入したコンテキスト解析手法である Ambiguity-based Context Analysis System (ACAS) の概要について述べる。ACAS はコンテキスト解析に曖昧さの概念を持たせることで、従来手法における問題を解決するシステムである。

ACAS の動作はサーバサイドを想定しており、データベースに保存されるセンサデータに対してあらかじめコンテキストのルールを定義する。そして、そのルールの中からアクチュエータで必要となるコンテキストを抽出する。ルールをあらかじめ複数定義することにより、コンテキスト解析の再利用性を高め、アルゴリズムの移植性を高める。以下に ACAS の機能要件を述べる。

- センサデータを曖昧な表現によって扱う

  センサデータを曖昧さを含めることで、3.1.1 章で定義したコンテキストの問題点のうち、センサデータの誤差、センサデータの不到達性、環境の動的な変化といったものを解決する。曖昧さを扱うための具体的な理論としてはファジィ理論を用いる。ファジィ理論のより詳しい説明は第 4 章で改めて行う。

- コンテキストの定義を言語化する

  コンテキストの定義を言語化して表現することで従来のコンテキストの表現で問題となっていた、コンテキストの表現方法やアルゴリズムの移植性といった点を解決する。曖昧さは人間の言葉に代表される自然言語と非常に相性が良い。そのため、コンテキストの表現に自然言語を用いることは理にかなっている。

3.3 ACASの想定環境

本節では、ACASの想定環境を図3.5を用いて説明する。ACASでは基本的に室内におけるコンテキスト解析を想定している。

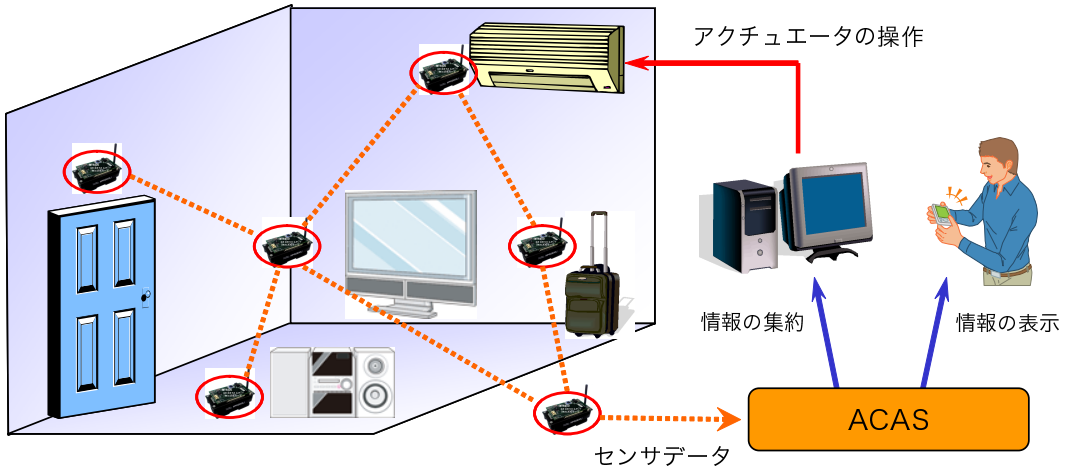


図3.5: 想定環境の概観

- 想定環境

  複数の高機能なセンサやコンピュータによって構築されるユビキタス環境ではなく、通常の部屋やオフィス、一般家庭などを対象とする。

- 使用センサ

  上記の想定環境に、温度や照度などの基本的なセンサデータを取得する小型センサを数十個程度取り付けて解析を行う。機能の限定されたセンサを利用する理由は、将来的にあらゆる場所にセンサを設置する場合には、高機能なセンサよりも安くて小型の汎用センサが用いられると考えられるためである。

- 対象コンテキスト

  汎用的なセンサを利用する場合には、特定のコンテキストしか取れないような解析手法ではなく、幅広いコンテキストを定義できるアルゴリズムが必要となる。室内空間であるので取得可能なコンテキストとしては、人の行動や環境の状態変化、機器の動作状況などが考えられる。それらの低次のコンテキストを複数取得して、より高次の空間コンテキスト情報に抽象化していく。

- ACAS の利用対象

  ACAS の利用者としては、システム開発者を想定している。具体的には、顧客の求めるシステムをハードウェアからソフトウェアにいたるまで、まとめて提供するシステムインテグレータのような開発者である。コンテキストアウェアシステムの構築時に ACAS は用いられると考えられるため、本システムは解析手法の提案であると同時に、ミドルウェアとしての役割も果たすと考えられる。

3.4 想定シナリオ

本節では、従来のコンテキスト解析手法で問題となる事柄に対して、ACAS によってどのような解決策がとれるのか、2 つのシナリオを挙げて考察する。

3.4.1 人の行動に関連するシナリオ

> A さんはユビキタスシステムの稼動している部屋にいる。ある日、B さんが A さんの部屋を訪れて自身の研究について話し合った。B さんは暑がりで、議論の途中にエアコンの温度を低くして欲しいと言った。A さんは今の温度をちょうど良いと感じていたが、B さんの要求する温度に変えた。

このシナリオにおいて最も重要な所は、快適さの基準となる対象を誰にするのかという点である。表 3.2 に従来手法における快適な温度の基準例を載せた。単純に最初に入室した人を優先する方法や、2 人の快適な温度の平均を求めてその温度に合わせるといった方法が考えられるが、その場合、対象人数が大きくなった場合や快適な温度に大きなばらつきが合った場合などでは全ての人が満足するような設定にすることができず、逆に不快に思うような人が多く出てきてしまう。また、ルールが複数存在する中で、いくつかの条件が重なったり同時に複数のイベントを検出した際に、正常なコンテキスト解析を行えるのかが問題となる。

このシナリオに対して、ACAS を用いたコンテキスト解析を行うことを考える。パラメータとして先に与えておくものは、各個人の快適な温度、アクチュエータでの優先度、イベントの優先度、などである。まず、個人の快適な状態をそれぞれが線形の値として持っていて、室内にいる人の中で最も優先度の高い人の重みを強くする。その上でイベントの優先度も含めた、全ての重みを合算して重心を求めることで、全ての人の好みを考慮した快適な温度を導き出せる。

表 3.2: 従来手法における快適さ (気温) の基準

基準とする指標	内容
全員の平均値	室内にいる人全ての快適な温度の平均に合わせる
最初に入室した人	元からそこにいる人の快適な温度に合わせる
部屋の所有者を優先	部屋の所有者となる人の快適な温度に合わせる
快適な温度が最も高い人	快適な温度が最も高く設定されている人に合わせる 室内にいる人のパラメータを常に監視する必要がある

3.4.2 環境の動的な変化に関連するシナリオ

環境の動的な変化に関連したものとして、前節のシナリオの続きとなるものを述べる。

> また、午前中は良い天気だったのにも関わらず午後には夕立が降り、室内の気温や照度が短時間で大きく変化した。突然暗くなったことで、照度センサの値で稼動していた入室検知システムと機器の使用状況を確認するシステムが正常に動作しなくなってしまった。システムを再起動している間、センサを利用したアプリケーションは一時的に利用することができなかった。

このシナリオでは、室内の気温や照度が短時間で大きく変化した際に、それまで稼動していたサービスをいかに維持させるかということが問題となる。従来の手法では、変化の幅の絶対値から環境の変化に対応させる方法や、環境変化の指標となるセンサデータを用意しておいて、それを基準として閾値の変更を行うなどの方法を取っていた。いずれも、環境に合わせてルールベースの条件の値を変更するということを行う。しかし、このような方法では現在起動しているサービスをそのまま持続して利用することはできず、一度データのリセットやシステムの再起動が必要になる。

そこで ACAS を利用することで、このような環境の動的な変化を考慮した上でサービスを継続したままでのコンテキスト解析が可能となる。ACAS では、環境の状態ごとに条件を定義するのではなく、曖昧さの定義によって環境の変化もコンテキスト解析のパラメータの一部となることができる。そのため、急に環境の状態が変化しても閾値を変更する必要はなく、常に変化の値を考慮した解析が行われる。

図 3.6 では従来手法による環境変化への対応と、同様の条件で ACAS を利用した際のコンテキスト解析方法を比べている。従来手法では、環境の状況ごとに閾値をあらかじめ定義ファイルなどで書き表す。そして、実際にシステムがそのような環境になった時に、閾値を切り替えてコンテキスト解析を行う。一方、ACAS では「晴れ」や「雨」などの環境の状態もファジィ集合によってパラメータ化しておくことで、通常のセン

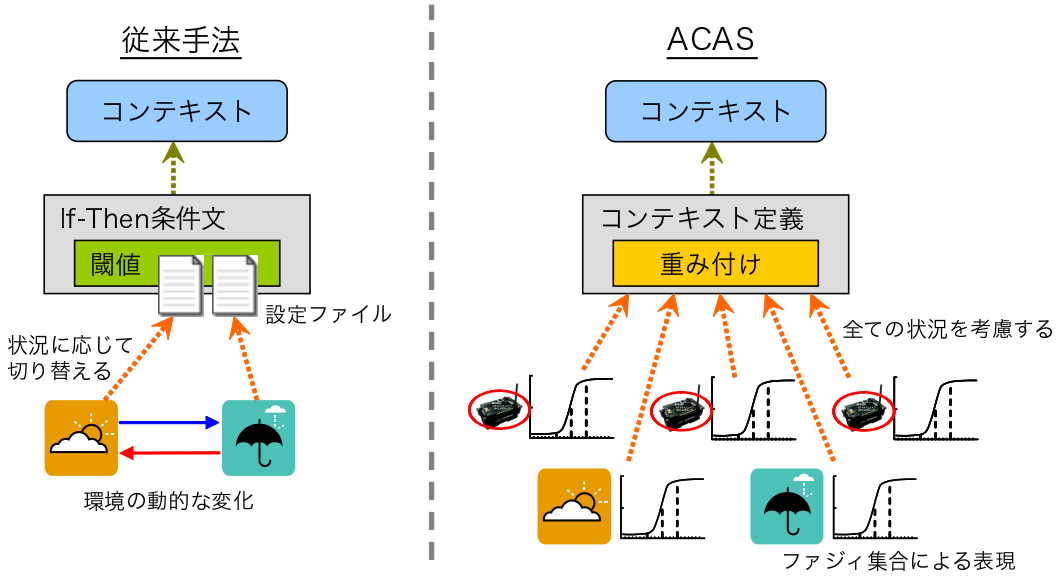


図 3.6: 環境の動的な変化への対応

サデータと同様に扱うことができる。そして、センサデータと共に環境の状況もコンテキストに対する重みにできるため、それらを合算してコンテキストの重心を求めることでコンテキストを得ることができる。コンテキストに関係すると思われる全ての事象を、同様のファジィ集合であらかじめパラメータ化しておけば、全てを包括してコンテキストを解析するため急な環境の変化にも対応することができる。

従来の手法と比べ、ACAS を利用することによってデータのリセットやシステムの再起動をする必要がなくなるため、よりシームレスなシステムの運用が可能になると考えられる。サービスを継続させたまま環境の動的な変化に適応させることで、アクチュエータの操作をよりむらなく行えるようになり、それがコンテキストアウェアシステム全体を効率的に動作させることにつながる。

3.5 関連研究との比較

本節では、関連研究を挙げて ACAS との比較を行う。比較するシステムの特徴を述べた後、ACAS との差異を明確にしてそれぞれの利点と欠点についてまとめる。

3.5.1 コンテキスト解析におけるメタ情報の利用

早稲田大学の藤波香織らは論文 [27] にてコンテキストアウェアシステムにおけるメタ情報の利用方法として、Java 言語で実装したフレームワークを提案している。彼らはコンテキストアウェアシステムの特徴である超多様性が、システム開発の問題であ

るとして、メタ情報とコンテキストとをどのように扱うべきかについて論じている。そして、そのフレームワークを適用したアプリケーションを開発し、コンテキスト定義の変更や更新、情報の追加が容易になることを示した。藤波らが提案するフレームワークと ACAS とを比較すると、メタデータを利用することで、上位コンテキストへの判断基準となる下位コンテキストをどのように定義しているのかが明確になり、システムの見通しが良くなり人間に理解し易くなることが分かる。しかし、彼らが提案するのはよりシステム全体をイメージしたフレームワークであるため、コンテキスト解析の精度や予想外の事象が起こったときにどのような解析が行われるか、など解析にのみ特化したような評価がされていなかった。結論としては ACAS のほうが分解能が高いため、より粒度の細かいコンテキスト情報が取得できる点で優位である。

3.5.2　ファジィ理論を応用したコンテキストのクオリティ表現

九州大学の久住憲嗣らは論文 [4] [19] にて、コンテキストアウェアシステムにおけるコンテキストの表現にファジィ理論を利用することを提案している。ファジィ理論によってコンテキストの表現が容易になると論じているが、その実装やどのような動作概要であるのかについてシステム内部に踏み込んだ言及がされていない。そのため ACAS と直接の比較は行えなかった。しかし、彼らの研究が提案の段階で留まっているのに対し、ACAS では実際にプロトタイプの実装を行っている点で優位である。センサデータをファジィ集合へ変換するモジュールや、ファジィ推論部のシステム構成など本システム独自の部分で明確な違いがあるため差別化を図ることができる。

3.6　本章のまとめ

本章では、従来のコンテキスト解析手法における問題点を挙げて、曖昧さを利用した解決方法について考察した。そして ACAS を利用することで状況が改善されると思われる想定シナリオを述べ、どのような環境で効果が期待できるのかを述べた。また、従来のコンテキスト解析手法の一つであるメタデータを利用したコンテキスト解析手法と比較して、ACAS における利点についてまとめた。その結果、コンテキスト解析の分解能の点で ACAS に利点があると考えられることを述べた。

次章では、曖昧さを表現するための理論である、ファジィ理論について詳しく解説を行い、実際にどのようにコンテキスト解析に利用するのかを論じる。

第4章　曖昧さの表現方法

本章では、コンテキスト解析に曖昧さを導入するための理論として、ファジィ理論について説明する。そして、ACAS においてコンテキスト解析とファジィ理論をどのように関連付けるのかを述べる。

4.1　ファジィ理論の概要

コンテキスト解析は、使用するセンサデバイスや扱うセンサデータの形式、アクチュエータでの動作などを考慮する必要がある。このような多様性のために、ルールベースに代表される 2 値理論で全てを解決するには不十分である。そのため第 3 章では、コンテキスト解析に曖昧さを持たせることを提案した。本節では、曖昧さを扱うための理論としてソフトコンピューティングの一つであるファジィ理論を説明する。

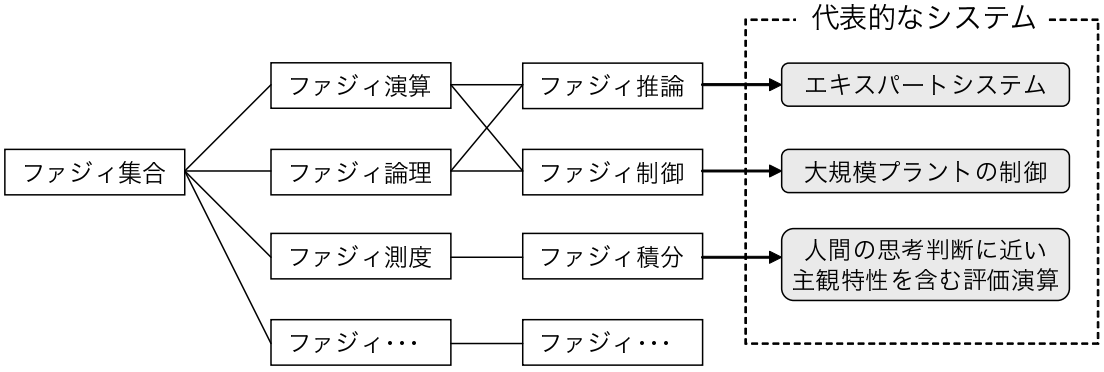


図 4.1: ファジィ理論の派生

まず、ファジィ理論についての概略を述べる。ファジィ理論とは、従来の確率論とは異なる曖昧さの表現やその処理の方法として提案されたものである。ファジィ理論そのものは 1965 年に L.A.Zadeh の提案した Fuzzy Sets (ファジィ集合) [16] から端を発した理論である。その後、現在までにファジィ推論やファジィ制御など、システムに曖昧さを導入するための理論として拡張され続けている。図 4.1 にファジィ集合を元として派生した理論のいくつかを載せた。実際には、ここに挙げたものよりもはるかに多くのファジィシステムが応用研究として提示されている。しかし、ファジィ理

論の肝となるのは、ファジィ集合とファジィ集合同士の演算であるファジィ推論のみである。一般的にファジィ理論を扱う際には、これらファジィ集合とファジィ推論を指すことが多い。ACAS においても、ファジィ推論による解析結果から、コンテキスト解析に曖昧さを加味する方法を取っている。

また、日本においてはファジィ理論についての研究が特に盛んであり、実際にファジィ理論によって得られる曖昧さを制御に組み込んだ電化製品が数多く発売されている。有名なものとしては、全自動ファジィ洗濯機というファジィ制御を組み込んだ製品がある。この洗濯機は、光センサで洗濯水の汚れを読み取りそれをファジィ推論を組み込んだ制御に渡すことで洗濯時間を変化させるものである。それ以外にも、掃除機、炊飯器、電子レンジ、ビデオカメラなど生活家電のいたるところにファジィ理論が応用されている。

4.2 ファジィ集合

ファジィ集合はファジィ理論全体の元となる考えであり、ファジィ理論を利用して曖昧さを表現する際には、全ての事象はファジィ集合によって表現される。一般的な集合の考え方としては、ある値がその集合に属するか否かの 2 値 (1, 0) のみを取る。この通常の集合のことをクリスプ (crisp) 集合という。クリスプ集合において、その値がある集合に属しているかどうかを表すものを特性関数という。これに対して、ファジィ集合は値に対してメンバシップ関数という線形グラフの数値によって集合を表す。

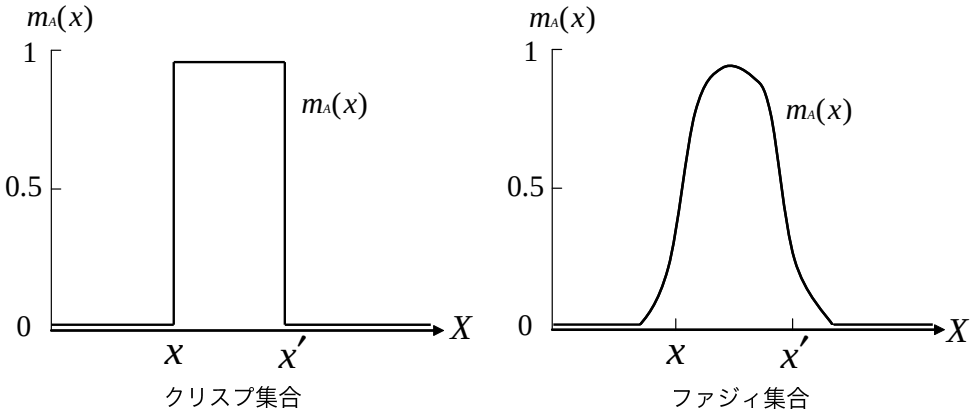


図 4.2: クリスプ集合とファジィ集合

図 4.2 に載せたのが、クリスプ集合における特性関数とファジィ集合でのメンバシップ関数である。特性関数では単に (1, 0) のどちらかの一方の状態でしかないため、その間 $(0 < m_A(x) < 1)$ で表わされる数値は意味を成さない。一方、メンバシップ関数は、その要素の度合いを示すため 0 に近いほどその度合いが弱く、 1 に近いほどその

度合いが強いことを表す。さらに詳しく見ると、クリスプ集合の場合は、 x から x' の範囲で $m_A(x) = 1$ の値を取り、それ以外は常に 0 となっている。クリスプ集合の特性関数は 0 か 1 のどちらかの値だけを取り、 X は x から x' の範囲で性質 $m_A(x)$ を満たすということがグラフから読み取れる。一方、ファジィ集合では、 X が x と x' のちょうど中間の位置で、 $m_A(x) = 1$ となり、それ以外の範囲では負の二次関数のグラフのように度合いが変化している。この増減の際の端数の重み付けによって、クリスプ集合とは異なる集合の定義が可能となる。もう一つより分かりやすい例として、図 4.3 に「暑さ」という現象に関するメンバシップ関数を示す。

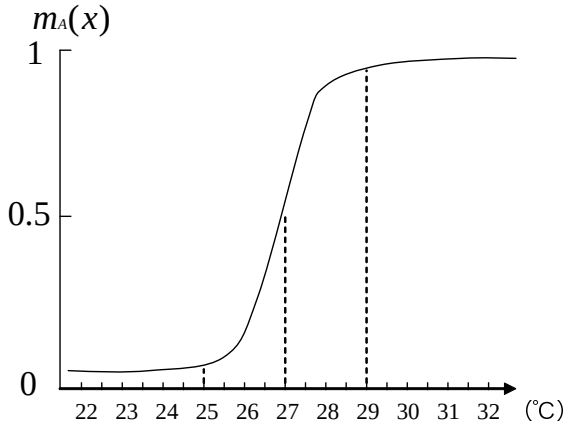


図 4.3: 暑さの度合いを表現するメンバシップ関数

このグラフでは、25 ℃から少しずつ度合いが増えていき、27 ℃で度合いが 0.5、29 ℃において 1 となっている。25 ℃以下のとき「暑さ」の度合いは 0 なので、メンバシップ関数によって表されるのデータの意味は「まったく暑くない」ということになる。同様に、27 ℃のときは「ほどほど暑い」、29 ℃以上で「暑い」を表現している。

このようにファジィ集合で事象を表現すると、度合いによってその状態がどのくらい強いものなのかを明示することができる。クリスプ集合では 2 つの状態しか表せないのに対して、ファジィ集合では度合いを無限に細かくできるため、人間が無意識的に重み付けを行っているような、曖昧な事象の表現が可能となる。コンピュータでは 1 と 0 による制御が行われているため、曖昧さを扱うためにはファジィ集合を利用した重み付けを行った上での論理計算が必要となる。

4.3 ファジィ推論

ファジィ推論はファジィ理論の応用のうちで最も重要なものと位置づけられる。ファジィ推論を発展させたのが、専門家の論理思考プロセスをコンピュータ上で再現した

エキスパートシステムである。また、一般的なルールベースによる条件判断は、ファジィ推論を 1 と 0 の 2 値によって判断した特別な事例であると考えられる。

ファジィ推論を説明するにあたり、理論の基礎となる三段論法について論じ、前件部と後件部、解の合成の方法、具体的なファジィ推論の実例という順で述べる。

4.3.1 三段論法

ファジィ推論を述べるために、まず三段論法について説明する。三段論法は古くはアリストテレスの提唱した推論方法であり、さまざまな応用理論へと発展しているが、その中身は非常に単純である。以下に簡単な例を挙げる。

- 鳥ならば空を飛べる
- 空を飛べれば向こうの島に行ける

このような 2 つの前提条件がある。ここで、「鳥ならば」= (A)「空を飛べる」= (B)「向こうの島に行ける」= (C) というように文を意味ごとに分けて簡略的に扱えるようにする。このとき、前提条件は次のように表せる。

鳥ならば空を飛べる	(A) → (B)
空を飛べれば向こうの島に行ける	(B) → (C)

このとき三段論法によって、

鳥ならば向こうの島に行ける	(A) → (C)

という推論が導かれる。一般的に、三段論法ではこの (A) (B) (C) の順番と真偽を逆にすることでさまざまな推論が可能となる。実際のファジィ推論では、この三段論法を次のような形式で利用することが多い。上記の鳥の例において、さらに次のような命題が与えられる。

- この動物は鳥である

「この動物は鳥である」という与えられた命題と、前提条件の一つ目「鳥ならば空を飛べる」から、「この動物は空を飛べる」という推論が成り立つ。

以上のことから、(A) → (B) → (C) という条件が成立しているときに、(A) → (C) を導き出す理論のことが三段論法の最も単純な形式であることが分かった。ファジィ推論はこの三段論法をもとにして、複数の条件から答えを導くものである。

4.3.2　前件部と後件部

本節では、ファジィ推論におけるプロダクションルールの定義について説明する。プロダクションとは推論を行うための「規則」のことで、プロダクションルールによってシステムは入力値に対する推論を行う。ファジィ推論でのプロダクションルールの定義は、自然言語によって表現されることが多い。次に簡単な例を挙げる。これは部屋の温度が高く、天気が晴れであるとエアコンをオンにするという条件を表す。

If room temperature is hot and weather is fine then air conditioner is on.

ここで、*If* から *Then* の間にはさまれている部分が前件部、*Then* 以降が後件部と呼ばれる。図 4.4 にプロダクションルールの定義イメージを載せた。実際は一つの条件文だけでなく複数のプロダクションを定め、それらを並列的に評価することで推論を行う。*“room temperature”*、*“weather”* はそれぞれファジィ集合によるメンバシップ関数で表現され、入力された値に応じた重み付けがされる。まず、前件部の度合いを求めそれらを足し合わせる。その後、後件部のメンバシップ関数から重心を求めることで推論値を導き出す。そして、結果として得られた *“air conditioner”* のファジィ集合を非ファジィ化し、実際の数値へと変換したものを受け取ることで、エアコンを何℃で動作させればよいのかが正確な数値と共に分かる。

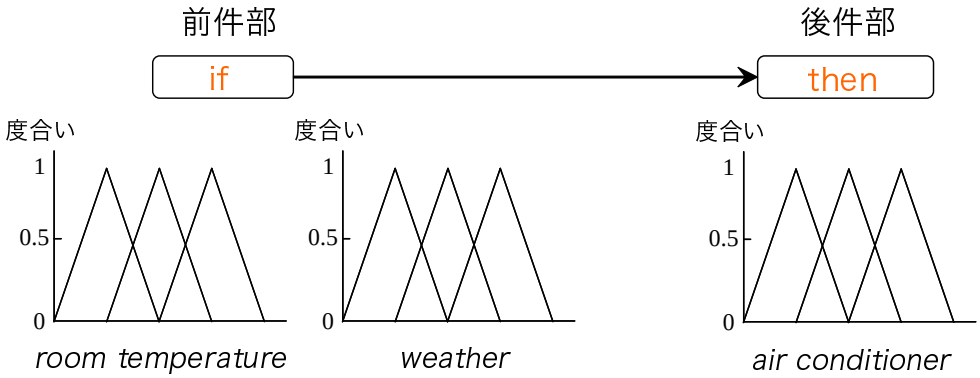


図 4.4: プロダクションルール

プロダクションルールは複数定義することができる。ファジィ推論の演算では、それぞれのルールから度合いとして得られたファジィ集合を合算して重心を求めることで複雑なルールから一つの結果を導き出す。このルールの複数定義こそが、ファジィ理論で曖昧さを表現する最も重要な部分である。これを通常のルールベースで行おうとすると、ベン図を何重にも組み合わせる条件の定義となるため非常に分かりにくく、また 2 値理論ゆえに条件の境界が明確となり曖昧さを扱うことが難しい。

4.3.3 推論手法

ファジィ推論を行う上で、プロダクションルールから解を合成して重心を求める作業が必要となる。その推論手法の代表的なものとして、Min-Max 重心法、代数積-加算-重心法、簡略化推論法などがある。

- Min-Max 重心法

  前件部の解を求める際には AND 演算を利用し、後件部の解には OR 演算を利用して推論を行う。つまり前件部ではメンバシップ関数の度合いが最も小さい値を取り、後件部では最も大きい値を取るアルゴリズムである。

- 代数積-加算-重心法

  前述の Min-Max 重心法では、Min 関数と Max 関数という非線形性の強い演算を使用しているため、推論結果が必ずしも直感に従ったものにならないことがあった。それを、Min 関数を代数積に、Max 関数を加算に置き換えたものが代数積-加算-重心法である。

- 簡略化推論法

  現在、ファジィ推論の主流になっている推論手法がこの簡略化推論法である。前件部には Min-Max 重心法における AND 演算を利用して行い、後件部のファジィ集合を実数値に置き換えて代数積-加算-重心法を適用したアルゴリズムである。後件部を実数で表現できるため、Min-Max 重心法に比べて計算量が少なくなりより早く演算結果を求められるようになる。

ACAS における推論手法は Min-Max 重心法を用いる。次節にて具体的な数値による演算の仕方を述べる。

4.3.4 ファジィ推論の具体例

ここでは実際にプロダクションルールを定義して、ファジィ推論の演算についてより具体的な例を見る。シナリオは以下のものである。

海外旅行中にとあるレストランに行き食事をした。チップをあげようと考えたが通常は食事代の 10 %から 20 %が相場であるという。このとき、店の全体としてのサービスの質と料理の質の双方から、最適と思われるチップの量を決めたい。

このようなシナリオではファジィ推論が大変有効である。通常のルールベースによる

条件判断よりもより自然な回答が求められる。以下に列挙するのが、このシナリオで定義するプロダクションルールである。

（1）*If service is poor and food is not delicious then tip is small.*
（2）*If service is soso then tip is average.*
（3）*If service is good and food is delicious then tip is big.*

これらのルールに対するメンバシップ関数を図 4.5 に示す。service は poor, soso, good の 3 つのファジィ集合、food は not delicious, delicious の 2 つのファジィ集合で表される。横軸はデータ入力時の扱いやすさを考え 0 から 10 の幅を持たせた。縦軸が度合いを表しており、度合いが 1 のときにその集合の特性を満たす。例えば、service が 3 のときは poor の度合いが 0.5、soso の度合いが 0.5、good の度合いが 0 というように複数の集合に属して表現される。

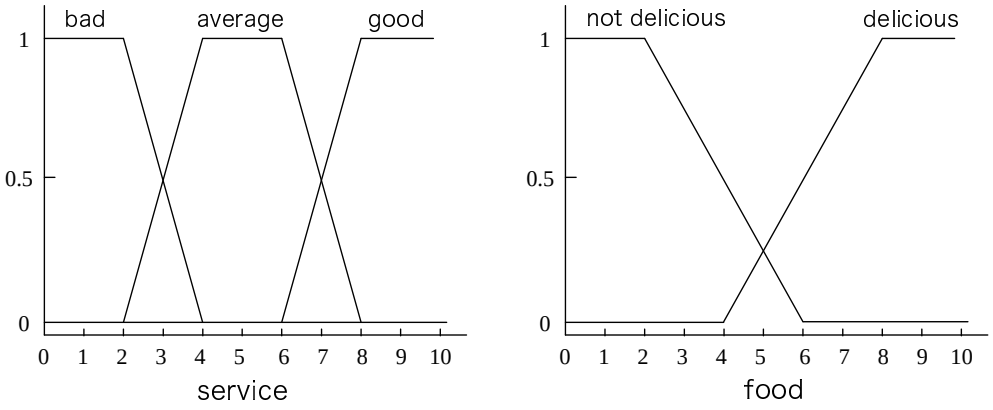


図 4.5: 前件部のメンバシップ関数

一方、後件部のメンバシップ関数を図 4.6 に示す。実際に求めたい値はチップを何パーセント払えばよいかということなので、横軸は 0 から 25 で単位は%である。

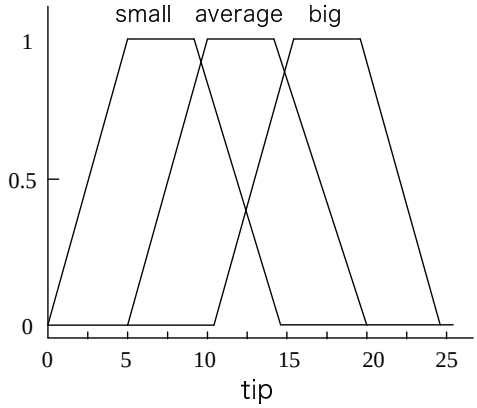


図 4.6: 後件部のメンバシップ関数

このとき、入力として「service = 5、food = 7」という値について推論を行う。図 4.5 から前件部に関して、service は soso の度合いが 1、good の度合いが 0 となり、food では delicious の度合いが 0.75 となる。他の集合に関しては全て 0 であり、直接推論に影響しないためここでは省略する。次にプロダクションルールを見ると、soso と delicious の関係する記述は

(2) *If service is soso then tip is average.*
(3) *If service is good and food is delicious then tip is big.*

である。これらのルールを元に前件部から後件部の線形グラフを得ると図 4.7 のようになる。ルール (2) から service が soso である時は、tip の average と対応していて、その度合いが 1 のため average は 0 から 1 の全ての範囲を取る。そして、ルール (3) から service が good で food が delicious である場合、tip の big に対応しており、前件部は Max 演算を行うため 0 と 0.75 で評価を行い 0.75 が得られ、後件部の big のファジィ集合を 0 から 0.75 の領域で切り取る。それら 2 つを重ね合わせ、得られた図形の重心を求めることで後件部の度合いとなる。この例では、結果として 12.5 という値が出力された。ゆえに、チップは料理代の 12.5 %分を払えば、サービスの質と料理の質の両方を考慮した最適なチップの量になる。

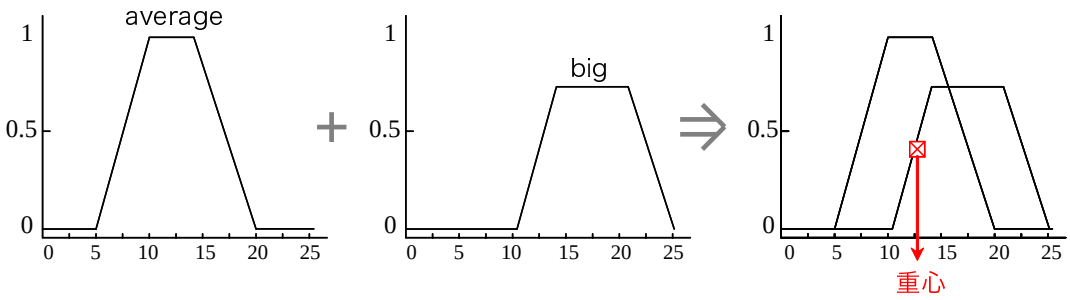


図 4.7: 解の合成

ただし、この例では (1) のルールが全く利用されていない。これはファジィ推論を利用する上で度合いが 0 の集合は重みを持たないため、計算処理を高速に行う目的から省かれている。また、実際にはこのような事象に対してプロダクションルールを定義するとき、ルールが 3 つでは少なすぎる。ルールが 3 つであればルールベースによる条件判断でも十分推論が可能であるため、ファジィ推論を利用する意味がなくなってしまう。今回は例として述べるために、単純なルール 3 つだけを示しただけであるが、本来であればファジィ集合として 10 個以上の要素を定義して、ルールも数十程度用意しなければ十分な推論結果とは言えない。また、計算に負荷がかかるため直線的なメンバシップ関数を定義したが、二次曲線による表現を用いればそれだけ細かな重み付けが可能になるため推論結果の信頼性も向上する。

4.4 ACASにおけるファジィ理論の利用

本節では、これまでに説明したファジィ理論をどのようにコンテキスト解析システムに組み込むのかについて述べる。具体的な手順としては、センサデータをファジィ集合によって表現し、あらかじめ定義されたプロダクションルールに従って、ファジィ推論を行うことでコンテキストとしての出力を得る。

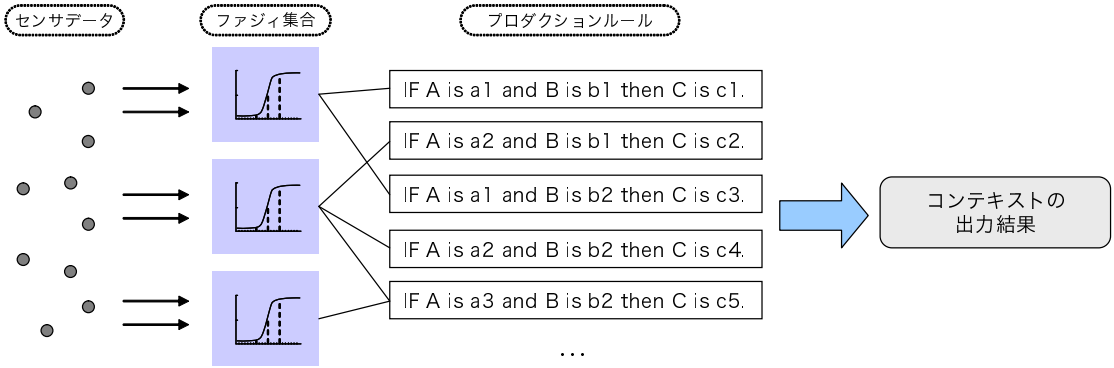


図 4.8: ファジィ推論を用いたコンテキスト解析の概略

図 4.8 にコンテキスト解析手順の概略を載せた。プロダクションルールは、まず一般的なコンテキストのベースとなるルールをあらかじめ定義した上で、ACAS の利用者が後からルールの追加を行えるような API を用意する。図にあるとおり解析を行えば、一回の推論で解析結果が得られる。しかしながら、コンテキストはセンサデータだけで判断されるものとは限らず、低次のコンテキストを組み合わせることで、高次の抽象化されたコンテキストを定義することがある。その際には図のように推論を一度しか行わない解析ではあまり意味が無いと考えられる。

ここで改めてコンテキスト解析について考えると、コンテキスト解析のフェーズは主に 2 つに分けられる。一つが生データから最も単純な形式のコンテキストへと変換する部分であり、ここでは「最小コンテキスト」と言うことにする。最小コンテキストはこれ以上コンテキストとして分割できず、粒度が最小であるコンテキストとなる。次のフェーズでは、一段階目で最小コンテキストとして変換されたものを、複数組み合わせてコンテキストを抽象化する。これを「抽象コンテキスト」と呼ぶ。この抽象コンテキストがシステム全体で最も多く生成され、アクチュエータなどにも利用される。それらのイメージを図 4.9 に示す。例えば、室内の環境を想定すると、最小のコンテキストとしてはモノの状態や人がいる・いないなどの情報である。そして最小コンテキストを元に抽象化したコンテキストとして、人が何をやっているのかや機器の動作状況、操作情報の把握などがある。そして、そのような抽象コンテキストをから室内全体におけるコンテキストを求めるというように、より抽象度の高いコンテキス

トへと変換するには解析を何度も繰り返すことが必要となる。また、一般に高次抽象化されたコンテキストのほうが空間的に広い範囲をカバーすると考えられる。

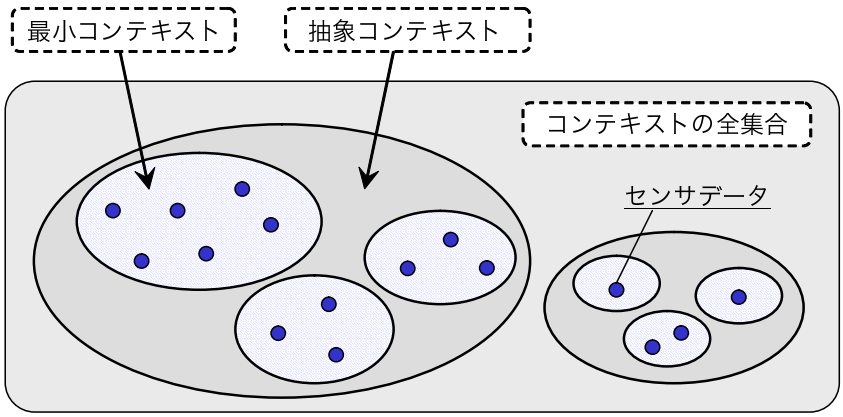


図 4.9: コンテキストの抽象化

そのように、高次コンテキストの抽象化を行うためには解析手法を何度か繰り返して行う必要がある。それはACASでも同様で、一度ファジィ推論によってコンテキストを得た後に、もう一度他のデータと共にファジィ集合化の段階から繰り返して解析を行うようにして抽象コンテキストを求められるような構造を持たせる。

4.5 本章のまとめ

本章では、システムに曖昧さを持たせるための理論であるファジィ理論について詳しく説明した。そして、解析にファジィ理論をどのように利用するのかを論じた。ファジィ理論の適用方法では、コンテキスト解析の段階ごとに数回利用することで、最小コンテキストから抽象コンテキストへと高次抽象化することを述べた。

ACASではファジィ理論を用いることで、コンテキスト解析に曖昧さを持たせているが、類似研究として久住らは論文 [4] [19] にて、同様にファジィ理論をコンテキストのクオリティ表現に用いているが、彼らはあくまでコンテキストの表現にファジィ理論が使えるという点に論を留めており、具体的にセンサを用いた実験やその評価を取っていない。その点で、本論文とは決定的な差異があると考えられる。

また本章では、参考文献として応用ファジィシステム入門 [21] とソフトコンピューティング用語集 [30] を、ファジィ理論について理解する上で、ファジィ理論を解説したウェブページ [2] [12] を参考にしたことをここに記す。

第5章　ACASの設計と実装

本章では、本論文で提案する曖昧さを考慮したコンテキスト解析手法 Ambiguity-based Context Analysis System (ACAS) の設計と実装について述べる。

5.1　ACASの概要

ACAS の特徴は、センサデータをファジィ集合で表して、重み付けによるコンテキスト解析を可能にすることである。まずはシステム全体の構造を述べた上で、センサデータのファジィ集合化と重み付けの手順に関する詳細を見ていく。

図 5.1 に実空間と ACAS のシステム全体像を模式的に示した。この図に関して、ACAS がどのような環境を想定しているのかを簡単に述べる。実空間の各部屋にそれぞれセンサが設置されている。センサの取得するセンサデータはネットワークを通じてサーバに送られ、データベース内に保管される。ACAS の動作場所としては、サーバサイド、クライアントサイドの両方が可能である。しかし、ある程度の規模のコンテキストアウェアシステムを考えたとき、より求められる形式はサーバサイドであるため、図ではサーバを含む ACAS の動作を示した。

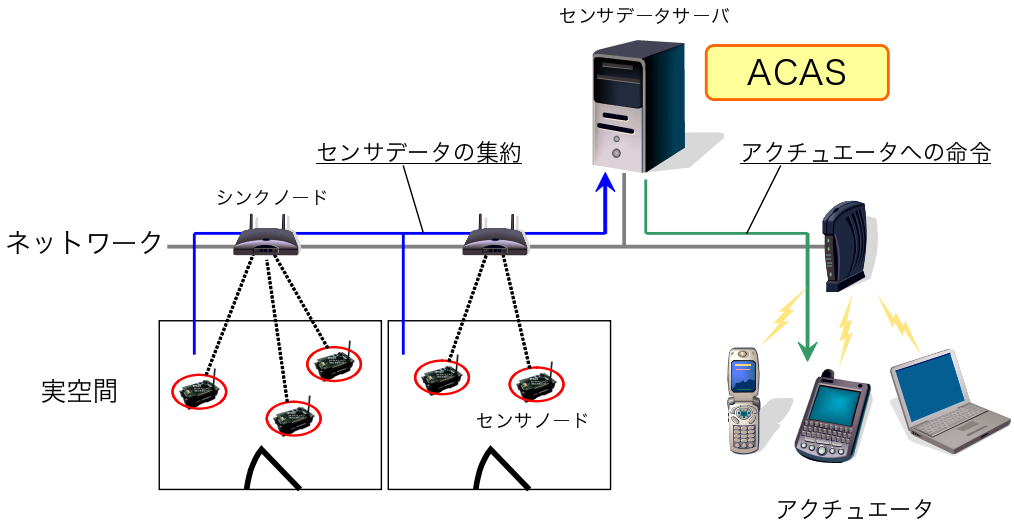


図 5.1: サーバサイドにおける ACAS の動作図

次に、どのようにセンサデータをファジィ集合化するのかについて述べる。センサデータをファジィ集合で表現し、重み付けを可能にするためには第4章で説明したプロダクションルールで用いられるメンバシップ関数が重要となる。ルールを構成するメンバシップ関数は、あらかじめ線形グラフがACAS内にデータとして定義されている状態である必要がある。そして、実際にそのルールの適応されるような条件になった時にセンサデータをメンバシップ関数の値へと変換してコンテキスト解析を行う。より詳細な事柄については、各モジュールの項にて改めて説明する。

コンテキスト解析後は、より高次コンテキストの一部として再び解析に回されるものとそのままアクチュエータに送られるものがある。ACAS内でどのようなコンテキストの定義を行ったかによって、その行き先は変化する。そして、アクチュエータによって実空間への動作が行われる。

5.2 ACASのシステム構成

本節では、ACASのソフトウェア構成について述べる。図5.2にACASのシステム構成図を載せた。ACASの核となるのは図中の「センサデータのファジィ化」と「ファジィ推論による解析」の部分である。各モジュールの詳しい説明に入る前に、まずは物理空間に関する状況、「センサデータの受信」について述べる。

ACASのシステムを利用する上で、物理空間におけるセンサ毎の配置方法や、センサの設置に合わせて必要なる情報はなく、センサノードや物理的な構成要素に対する制限を受けない。センサノードやデバイスに依存した違いは、ACAS内部で抽象化されるため考慮する必要はなく、特定のセンサノードに依存した解析も行わないため、状況に限定されず解析を行える。しかし、コンテキスト解析のルールを記述する上では、あらかじめセンサデータとコンテキストの状態とを把握しておく必要がある。ある数値データの特徴と、それに対応するコンテキストの状態を結びつけることは、エキスパートシステムに代表される、人間の感覚によって状況判断をルールとして定義するための前提となる。ただし、本当に正確な数値と状態の定義が求めれれるわけではなく、おおよそのセンサデータの値とその状態とが分かればACASで解析することができる。以降では、各モジュールの役割をデータフロー順に説明する。

センサデータのファジィ化

「センサデータのファジィ化」の部分では、センサノードから受信したデータをファジィ推論による解析可能な形式であるファジィ集合に変換する。外部からセンサデータ形式の定義ファイルを読み込むことで、後からセンサノードの追加も可能となる。

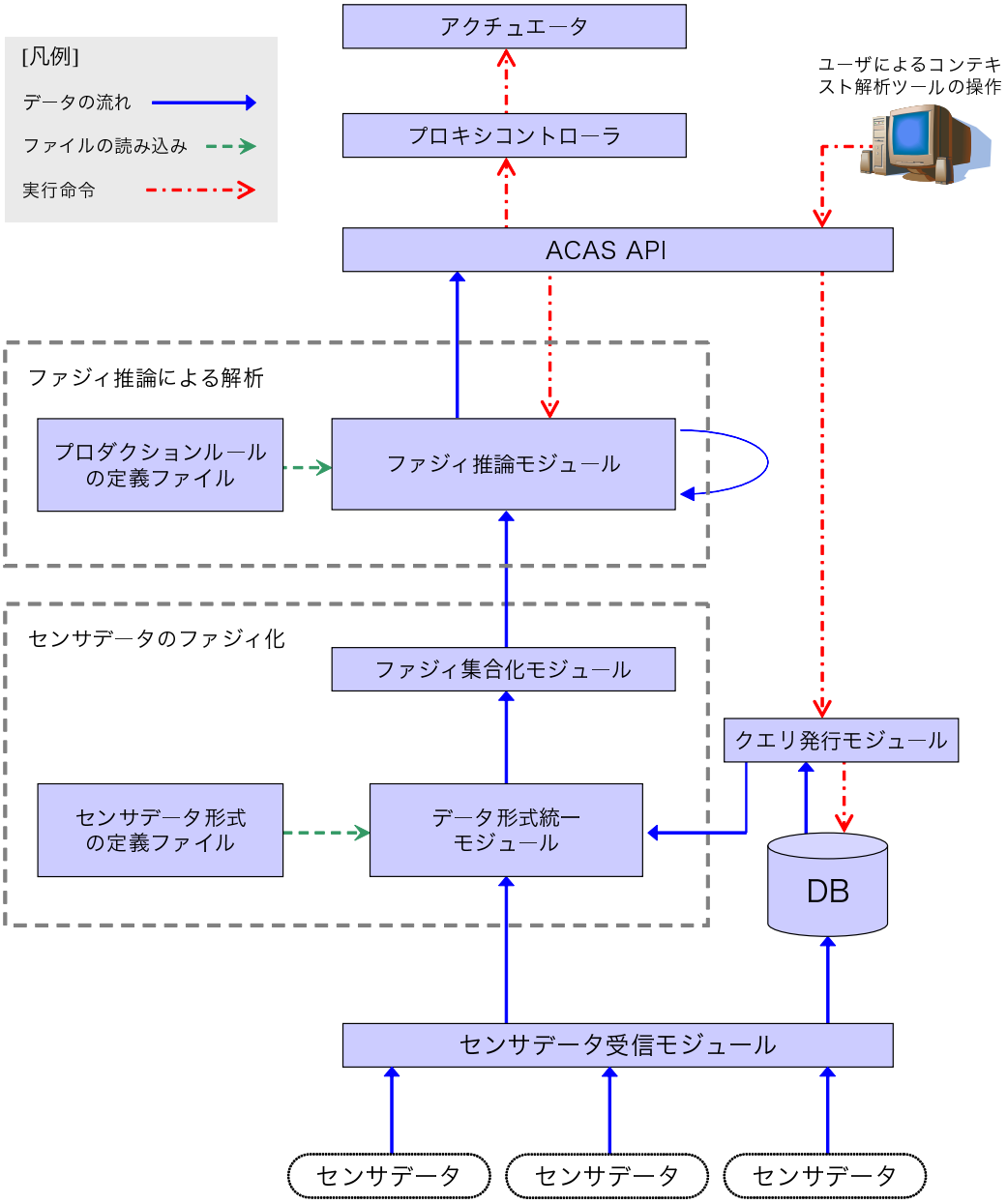


図 5.2: ACAS のシステム構成図

- センサデータ受信モジュール

  センサノードの種類ごとに、それぞれセンサデータを受信するための TCP port を用意する。後からセンサノードが追加されることも考慮に入れて設計を行う。通常、センサデータはデータベースに保存されるが、生データによるコンテキスト解析を行う場合にはそのままデータ形式統一モジュールにデータが渡される。

- データ形式統一モジュール

  センサデータは各センサデバイスごとに取得可能な値に差がある。光センサの値として 0 から 255 でデータを取得できるものもあれば、0 から 65535 を数値として返すもの、負の値を取るものもある。このようなセンサデバイスごとの差を、このモジュールにて正規化する。

- クエリ発行モジュール

  ユーザが直接入力を行うのは ACAS API に対してだけである。このモジュールでは ACAS API からの要求を受けたら、データベースにデータを要求するためのクエリを作成し、データの受け渡しを行う。そして得られたセンサデータをデータ形式統一モジュールに送る。

- ファジィ集合化モジュール

  センサデータ形式統一モジュールを経て、正規化されたセンサデータをこのモジュールにてファジィ集合に変換する。この線形グラフはシステムの演算効率を考え、曲線ではなく直線を複数組み合わせたもので表現する。

ファジィ推論による解析

ファジィ集合化されたセンサデータに対して、ファジィ推論によって解析を行いコンテキストを得る。推論を行うためのプロダクションルールは、外部の定義ファイルを読み込むことで入力する。ルールの記述は自然言語による定義となる。

- ファジィ推論モジュール

  ファジィ推論モジュールでは、プロダクションルールに従ってファジィ集合化されたセンサデータからコンテキストを求める。その際、コンテキストの重み付けや重心の求め方などには、ファジィ理論の応用としていくつかの手法があるが、ACAS においては第 4 章で説明した Min-Max 重心法を用いた。また、抽象コンテキストのように複数のコンテキストを合わせて、より高次のコンテキストを得る場合などを考慮した設計にする。

アクチュエータでの動作

ファジィ推論による解析が行われた後、出力されたコンテキストとアクチュエータの動作を関連付けする必要がある。そのための方法として ACAS API を用意する。実際にアクチュエータを操作するのは、API からプロキシコントローラを介する。

- ACAS API

  この API は ACAS でコンテキスト解析を行う上で、システム開発者がより効率的にシステムを運用できるようにするための関数の集合体である。次に列挙したものが API として提供するメソッドの一部である。

  ファジィ推論への命令関数 ／ プロダクションルールの定義関数
  コンテキスト解析結果取得関数 ／ データベースへの要求関数

  これらの関数を利用することで、システム開発者は ACAS を利用したシステムの開発が可能となる。

5.3 ACASの動作概要

本節では、ACAS で特に重要なモジュールに関する動作概要を述べる。以下に列挙したモジュール間の動作の内、a、b、c はセンサデータのファジィ集合化に関する手順であり、d はファジィ推論の動作に関するものである。

a. データベースとの接続
b. データ形式の統一
c. センサデータのファジィ集合化
d. ファジィ推論とプロダクションルール

5.3.1 センサデータのファジィ集合化までの流れ

ここではセンサデータを受信して、データベースに保存する部分、そしてデータの統一とファジィ集合化の手順について述べる。

a. データベースとの接続

本システムではデータベースに PostgreSQL を用いた。以下にデータフィールドの定義と実際のデータの値の一部を載せる。各センサデータは表 5.1 にあるような形式でデータベースに保存される。表 5.2 にはデータベースのサンプルデータを載せた。

表 5.1: データフィールドの定義

項目	データ型	項目	データ型
id	serial	voltage	real
xbridge id	integer	temperature	integer
upart id	integer	movement	integer
sensor id	integer	light	integer
seq	integer	timestamp	timestamp
field strength	integer	payload	–

表 5.2: データベースのサンプル

id	xbridge id	upart id	sensor id	seq	field strength	左下に続く
82657657	2	308	328154	6	0	
82657305	2	308	328154	58	0	
82657364	2	308	328154	60	0	

voltage	temperature	movement	light	timestamp
2.8	28	0	244	2007-08-10 00:02:40
2.8	28	34	244	2007-08-10 00:00:20
2.8	28	58	244	2007-08-10 00:00:44

b. データ形式の統一

データベースから取得したデータを、データ形式統一モジュールで正規化する手順を述べる。図 5.3 に 2 つのセンサデータ形式についての例を挙げた。図中の上にある帯状のものがセンサデータの幅を表し、下の 0 から 10 までの数値が正規化後のセンサデータの値となる。ここで正規化された値を 0 から 10 にした理由は、後のファジィ集合化の際に度合いが小数となるため、その時に扱いやすい数値にするためである。

センサデータ A の場合では、センサデータの数値の幅そのものを比率として 0 から 10 に変換すれば良い。センサデータが 170 であったならば、およそ 6.5 という値に変換される。一方、センサデータ B では少し特殊な事例を挙げた。ここではセンサデータの幅は 0 から 65535 の値を取るが、センサの特性として最小値が 0 とならないことがある。つまり、照度センサなどに対して黒い布などでセンサを覆いまったく光が入らない状態を作り出しても、その値が 0 にならない時がある。このようなセンサデータ

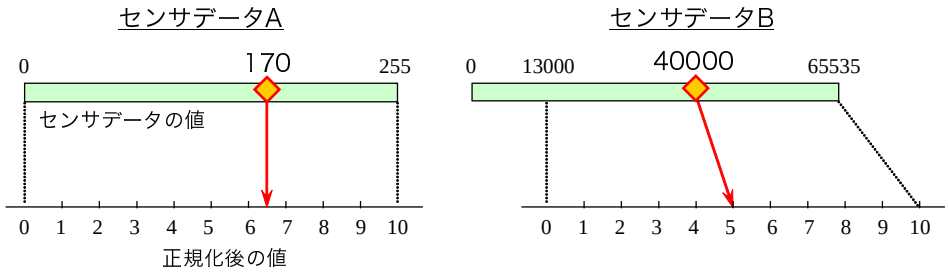


図 5.3: センサデータ正規化の概略

については、図 5.2 にあるセンサデータ形式の定義ファイルによりデータの値の特徴を記述して正しくデータの変換を行うようにする。この例ではセンサデータの 13000 を 0 に割り当てて、それを基準として割合を求めることにより値の変換を行っている。ゆえに、40000 というセンサデータは 5 に変換される。

c. センサデータのファジィ集合化

センサデータをファジィ集合へ変換するには、線形グラフで表されたファジィ集合を用意する必要がある。そして重み付けをするために、センサデータの入力関数でデータを登録する。データを登録するとその値から重みが求められる。図 5.4 のような少し複雑なファジィ集合から値を定義する手順を見ていく。入力関数を用いて

ACAS.inputSensorData (4.6);

というように入力すると、図の横軸 4.6 の値で各状態の度合いが判断される。この結果、各状態の度合いが表 5.3 のように求められる。

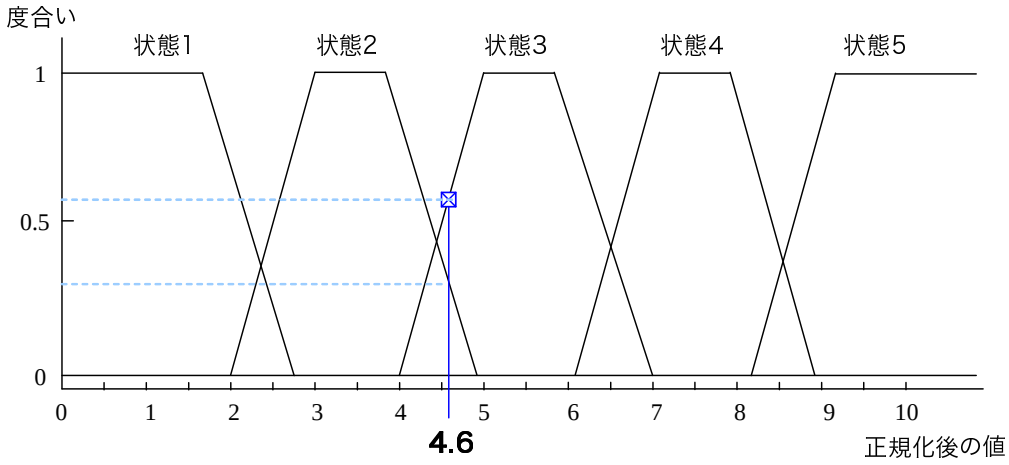


図 5.4: ファジィ集合化の概略

表 5.3: 重み付けの結果

	状態 1	状態 2	状態 3	状態 4	状態 5
度合い	0	0.4	0.6	0	0

表 5.3 では 5 つの状態のうち、3 つが度合い 0 となるため、実際にプロダクションルールでその状態について記述されたとしても結果には影響せず、状態 2 と状態 3 の重みからコンテキストが判別される。

5.3.2 ファジィ推論の動作手順

次に、センサデータからファジィ集合に変換した値についてプロダクションルールに則った推論を行うための手順について述べる。

d. ファジィ推論とプロダクションルール

ファジィ推論では、プロダクションルールの定義がコンテキスト解析のすべての要因となる。一般的なルールベースによる解析との一番の違いは、ファジィ推論では記述されたルールのうち、関係する全ての事象についてファジィ集合による度合いから推論を行うということである。また、ルールの記述には If-Then else のような明確な条件として定義するのではなく、ルールとして全ての条件を羅列するように記述する。表 5.4 に載せたのは、「賑わい」というコンテキストに対するプロダクションルールの一部である。これは、ファジィ推論に固有のルール定義の方法であり、ACAS の実装においてもこの記述方法を継承している。

表 5.4: プロダクションルールの例

if now-move is small then crowd is small.
if min-move is small then crowd is small.
if min-move is many then crowd is many.
if min-move is many and now-move is many then crowd is many.
if min-move is many and now-move is small then crowd is soso.
if min-move is small and now-move is small then crowd is small.
… … …

表 5.4 に載せたルールは非常に限定されたセンサデータのみを利用して、コンテキスト解析を行う場合の条件であるため、要素とその状態が 3 つしか表現されていない。そのため 15 ほどのルールで全ての状態を記述することができる。実際にはもう少しルールに記述される要素数が多くなるため、プロダクションルールも複雑になる。

前節の図 5.4 では状態 1 から状態 5 というような記述がされていたが、これはセンサデータやコンテキストの状態をシステム設計者が各項目ごとに定義することになる。例えば、温度などの場合ある温度では「寒い」「快適」「暑い」などの状態を言語化して定義する必要がある。

5.4 ACAS の実装

本節では、ACAS のプロトタイプの実装について述べる。

5.4.1 実装環境

本来、ACAS はサーバサイドで動作させることを目標としてきたが、本論文におけるプロトタイプ実装では、個人の PC 上で動作させた上でコンテキスト解析を行うシステムを構築した。システム構成は図 5.2 で述べた構成となる。実装に使用した PC の概要を表 5.5 にまとめた。ACAS の実装は Java 言語を利用した。

表 5.5: 実装に使用する PC の概要

名称	値
モデル	ThinkPad T41p(2373-GJJ)
CPU	Intel Pentium M 1.69GHz
メモリ	1.0GB
OS	Windows XP SP2
Java	1.6.0_03

また、センサデータは慶應義塾大学徳田・高汐研究室の実験環境である Smart Space Laboratory における実データを利用した。Smart Space Laboratory にはセンサデバイス uPart が室内に取り付けられていて、壁、プロジェクタ、机、イスなどミーティングルームにおける主要なモノの動作状態がセンシングされている。そのため、ACAS によるコンテキスト解析を行うには最適であると考えた。

5.4.2 ソフトウェア構成

本システムのソフトウェア構成について述べる。実装言語としては Java を利用したため、他のライブラリも基本的には Java との相互接続性のある言語を用いている。データベースとの接続には JDBC を利用し、次に述べるファジィ推論エンジンも Java によって記述されたものを選んだ。

ファジィ推論エンジン

ファジィ推論については関連研究として挙げた文献 [21] [30] にも述べられているように、先行研究が豊富にある。そのため、ファジィ推論のプログラムは一般に公開されているものを利用した。GNU General Public License で配布されている、FuzzyEngin [2] というライブラリを使用した。本論文の末尾に、FuzzyEngin の例としてファジィ推論を行うサンプルコードを載せた。このサンプルコードでは、レストランでの食事の際に料理のおいしさとサービスの質から、チップの量を決めるという事象に対して、ファジィ推論によって答えを導き出している。

ファジィ集合の定義

以下に、プロトタイプで実装したファジィ集合の定義を載せる。正規化したセンサデータの値をファジィ集合にする際に、表 5.6 のような識別名とその度合いの定義名を定めた。この表にあるデータの識別名と度合いの定義というのは、プロダクションルールの記述に利用する単語となる。例えば、

If temperature is small then CONTEXT 1 is XXX.
If movement is aberage and light is big then CONTEXT 2 is YYY.

というような形式でルールを構成する。ここに示したファジィ集合が、全てのコンテキストの根幹となる。また、これらのセンサデータの実際のメンバシップ関数は図 5.5 に示すような定義を与えた。

表 5.6: プロトタイプのファジィ集合定義

センサデータの種類	データの識別名	度合いの定義
温度	temperature	small / average / big
振動	movement	small / average / big
照度	light	small / average / big

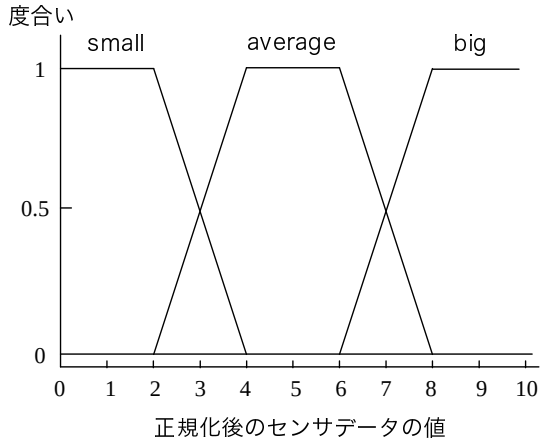


図 5.5: プロトタイプのメンバシップ関数

ここで、第 6 章の評価では、室内でミーティングが行われているかどうかについて比較実験をするので、それに関連するコンテキストの定義を載せる。まず、ミーティングというコンテキストを構成する要素としてはプロジェクタが ON であるということが必須であるのでそれをコンテキストと考える。同様にして机・イス・照明などの状況をコンテキストとして定義する。それらを表 5.7 にまとめた。

表 5.7: プロトタイプのコンテキスト定義

コンテキストの種類	コンテキストの識別名	度合いの定義
プロジェクター	projector	ON / OFF
机の揺れ	desk	vibrate / not vibrate
イスの揺れ	chair	vibrate / not vibrate
照明の点灯	room-light	bright / not bright
壁に光が当たる	wall	bright / not bright

以上のファジィ集合の定義から、プロダクションルールを定めることで室内におけるミーティングというコンテキストの定義を行う。ルールの基準は、定義する人間がミーティングをどのように認識しているのかによって異なる。今回のプロトタイプでは、一般的な人が考えるであろう状態をルールとして定義した。つまり、プロジェクターが ON で机・イスが動いており、照明が点灯しているということをミーティングが行われている場合の判断基準とする。具体的なプロダクションルールを表 5.8 に挙げた。実際には、机・イス・照明・壁は対象が複数であるため、そのセンサごとに定義する必要があるがここではまとめて記述している。

表 5.8: ミーティングのプロダクションルール

if projector is ON then meeting is on.
if projector is OFF then meeting is off.

if desk is vibrate then meeting is on.
if desk is not vibrate then meeting is off.

if chair is vibrate then meeting is on.
if chair is not vibrate then meeting is off.

if room-light is bright then meeting is on.
if room-light is not bright then meeting is off.

if wall is bright then meeting is on.
if wall is not bright then meeting is off.

ここに挙げたプロダクションルールに基づいてファジィ推論を行い、ミーティングのコンテキストを得る。コンテキストは重み付けされた形式で出力されるため、アクチュエータにその値を渡すことでよりきめ細かな機器操作を行えるようになる。

5.5 本章のまとめ

本章では、ACAS の設計と実装について述べた。本システムで最も重要であるファジィ推論を中心に、センサデータ形式の統一や、センサデータをどのようにファジィ集合に変換するのかを論じた。そして、各モジュールについてそれぞれの役割と、動作概要をまとめた。ACAS は基本的にはサーバサイドで動作するような利用を考えているが、コンピュータ単体に組み込むことでスタンドアロンな動作も可能である。プロトタイプの実装では、クライアント側で動作するシステムの実装を行った。次章では、プロトタイプの評価と実証実験について述べ、ACAS の性能について考察を行う。

第6章　ACASの評価

本章ではACASの評価を行い、その結果について考察する。定量的評価では、実際にコンテキストアウェアシステムを構築して、アルゴリズムの部分でACASとルールベース手法を切り替えることにより評価を行う。定性的評価では既存の手法との性能的な差を、コンテキストの拡張性やアルゴリズムの移植性などから考察する。また、公共空間にて行ったACASの実証実験についての詳細も記述する。

6.1　定量的評価

定量的評価を取るにあたって、サンプルとなるコンテキストアウェアシステムを構築した。アルゴリズムの部分でACASとルールベースと切り替えることで、それぞれの性能を比較する。そして、いくつかの指標を用いてどちらの手法がどのような点で優れているのかを明確にする。本節では、定量的な評価を行う上での評価環境や評価軸について述べた後、実際の評価結果についてまとめる。

6.1.1　評価環境

ACASの評価に使用したPCは、第5章の表5.5で述べたものと同等の機器である。評価のためのセンサデータとして、第2章で詳解したSmart Space Laboratoryの実データを入力値として利用した。設置されているセンサノードはuPartである。環境が動的に変化する状況をSSLabで測定された実データを変化させることで再現する。利用するデータは、夏のミーティング中に測定したセンサデータと、冬の同条件下におけるセンサデータである。コンテキスト解析アルゴリズム側では、受け取る光センサの値と温度センサの値が急激に変化することになる。評価に利用したセンサの設置場所とその個数は以下のとおりである。

プロジェクタ：1個　イス：4個　机：3個　壁面：1個

図6.1にSSLabにおけるセンサの設置場所を示した。これらのセンサ配置は、ミーティングのコンテキストと特に関係があると考えられる場所やオブジェクトから選定した。

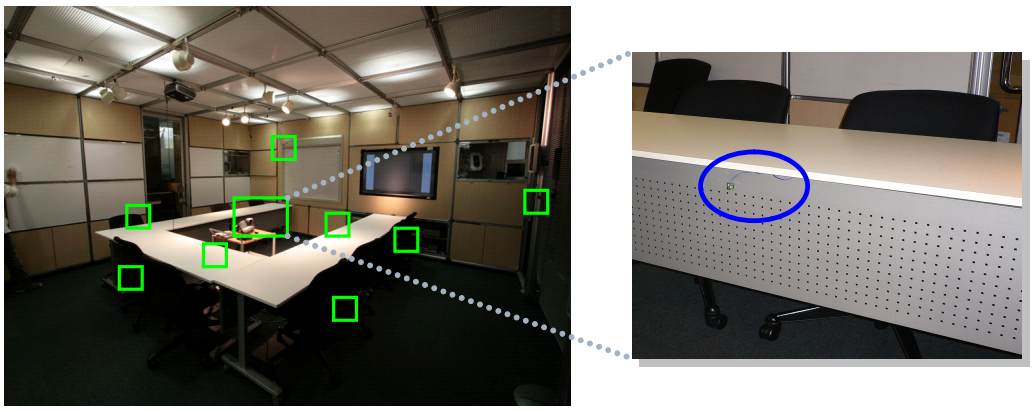


図 6.1: SSLab のセンサ設置状況

6.1.2 評価シナリオ

次に評価を行うためのシナリオについて述べる。

限られた機能しか持たないセンサが設置された環境において、その部屋の空間のコンテキストを取得したい。取得するコンテキストとしては、その部屋でミーティングを行っているのかどうかである。しかしそんな時、天気が晴れからどんよりとした曇りへと短時間の間に急激に変化した。

このシナリオの要点は短時間で環境が急激に変化することであり、それまで正しくコンテキスト解析できていた状態のものが、状態変化後も正しく動作可能であるのかが計測の基準となる。

6.1.3 評価軸

動的な環境変化への対応

正解となるコンテキストを明示した上で、環境の動的な変化にどれだけ追従できるかを調べる。動的な変化というのは、センサデータを擬似的に変化させることで発生させる。どのくらいの割合で正確にコンテキストを認識できるか、またはほとんど閾値から外れてしまうのかなどを確認する。

6.1.4 解析手法

次に、評価シナリオにおける各コンテキスト解析アルゴリズムの構築方法を述べる。ルールベースでは記述のためのルールを列挙し、ACAS ではコンテキストを表現するプロダクションルールについてまとめる。

ルールベース手法

ルールベース手法において、室内のコンテキストとしてミーティング中であるかという状態を定義するために、次のようなルールを与えた。これらのルールについてIf-Then文を用いてセンサデータの閾値を決めて解析を行った。

・プロジェクタが ON の状態である
・室内のいくつかのオブジェクトが動作状態にある
・室内の照明器具が点灯している

ACAS

ACAS におけるミーティングのプロダクションルールは、第 5 章の表 5.7 と表 5.8 にて定義している。そのためここではプロダクションルールの要点だけを書き出す。

・プロジェクタが ON であれば、ミーティング中である
・プロジェクタが OFF であれば、ミーティング中ではない
・机 / イス が振動していれば、ミーティング中である
・机 / イス が振動していなければ、ミーティング中ではない
・照明が点灯していれば、ミーティング中である
・照明が点灯していなければ、ミーティング中ではない

6.1.5 評価結果

定量的評価を行った結果について述べる。比較にあたり、夏のセンサデータに対してミーティングのコンテキストが判別できるようにルールベースと ACAS を初期設定した。その上で夏のセンサデータと冬のセンサデータを解析したときに、コンテキストの正答率がそれぞれどの程度になるのかを測定した。その結果を図 6.2 に載せる。試行回数は 10 回で、縦軸はミーティングのコンテキストを正しく判別できた回数を示す。各指標とも左側の棒軸が夏のセンサデータを与えたときの値で、右側の棒軸は冬のセンサデータを与えたときの値である。この図からルールベース手法も ACAS も夏のセンサデータに対しては 9 割の正答率であるため、非常に高確率で解析できている。一方、冬のセンサデータに対してはルールベース手法では 2 割、ACAS では 6 割という正答率であった。この結果から、環境が大きく変化したときにルールベース手法ではコンテキスト解析の正答率がそれまでの約 23 %まで低下して十分な解析結果とは言えないのに対して、ACAS では正答率はそれまでの約 65 %となっているため、環境の急激な変化により対応できていると言える。

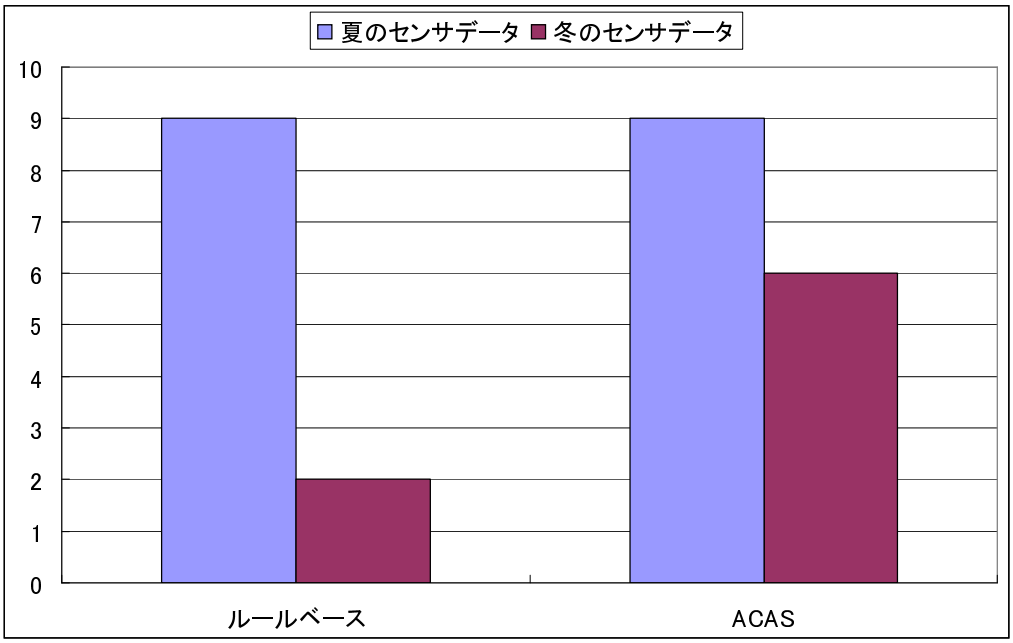


図 6.2: ミーティングのコンテキストの正答回数

6.2 定性的評価

定性的評価では、ACAS との比較対象として従来のルールベースによる手法と、メタ情報を利用した手法の 2 つを挙げる。以下に、それぞれの特徴をまとめる。

ルールベースによるコンテキスト解析手法では、センサデータやタグ情報などを If-Then 文によって条件分けをする。ルールベースではそれらの条件を組み合わせることでコンテキストを定義している。対象となるコンテキストによって、非常に複雑な実装を取ることもあるがここでは一般的なルールベース解析のみを仮定する。一方、メタ情報を利用したコンテキスト解析手法は、藤波らの論文 [27] に代表されるように、コンテキストの定義とハードウェアの情報をメタデータとして管理するものが多い。ハードウェアの情報というのは、センサデバイスの取得できるデータの種類や属性などである。アルゴリズムと実装を分離できるため、ハードウェアに依存しないコンテキスト解析が可能である。次に、定性的評価での検証項目を列挙していく。

- **複雑なコンテキストにおける拡張性**

  複雑なコンテキストとは、複数のセンサデバイスを利用した環境下で、何十もの条件の組み合わせによって表現されるコンテキストのことである。複雑なコンテキストが既に定義されている状態で、新たな条件の追加は容易であるか、部分的にコンテキストの表現を変更できるかなどを評価対象とする。

- **アルゴリズムの移植性**

  移植性とは、既存のシステムからアルゴリズムだけを切り離して、別のアプリケーションで直ちに利用可能であるかについての指標である。これは、センサデータの形式などに依存していると、うまく移植することができず再利用性の低いアルゴリズムとなってしまう。

- **コンテキストの強さの表現**

  コンテキストの強さとは、コンテキストの状態に対する度合いのことである。一般に考えられる、コンテキストの起こる確率とは異なる。例えば、室内のにぎわいをコンテキストとした場合、にぎわいのコンテキストに対して、「すごく」「まあまあ」「あまり」「まったく」などの表現が考えられる。

- **設定の容易さ**

  コンテキストアウェアシステムは初回構築時に設定を行う必要があるが、その設定の容易さや、不規則な環境下でも設定を行えるかの指標である。新たに条件を加える際の手間や、コンテキストのメンテナンス性などもこの指標に含める。

表 6.1: 定性的評価の結果

項目	ルールベース手法	メタ情報利用	ACAS
複雑なコンテキストにおける拡張性	△	○	△
アルゴリズムの移植性	△	○	○
コンテキストの強さの表現	△	×	○
設定の容易さ	×	○	○

表 6.1 から、ACAS は他の手法に比べ十分な強みを持っていることが分かる。複雑なコンテキストにおける拡張性では、メタ情報利用がコンテキストの論理記述できる点から最も優位であると考えられるが、ACAS でもプロダクションルールの追加によって対応できる。それ以外の項目では、ACAS は他の手法と同等かそれ以上の性能を持つ。これは ACAS がファジィ集合という人間の思考プロセスに近い表現によって推論を可能にしていることが影響している。コンテキストの強さの表現などは、ファジィ集合が得意とする部分であるので、従来のコンテキスト解析でルールを複数重ねて定義していた強さの表現は、ファジィ理論を使うことで非常に簡単に表現することができる。ACAS が従来の解析手法との一番の違いは、重み付けが容易に可能になる点であり、それが要因となってこのような評価結果となることが分かる。

6.3 ACASの実証実験

想定環境における他の解析手法との比較実験だけでなく、公共空間において ACAS の評価実験を行った。この実験は、ACAS の完全なシステムとしての評価ではなく、ファジィ推論によるコンテキスト解析部に重きを置いたものである。

6.3.1 実証実験の概要

慶應義塾大学湘南藤沢キャンパスでは、SFC Open Research Forum (ORF) という、大学での研究成果を発表するための催しが毎年行われている。本年度は、2007 年 11 月 22 日と 23 日の 2 日間、六本木ヒルズのアカデミーヒルズ 40 にて開催された。本論文もデモ展示を行い、ORF 会場にて ACAS を動作させた。そのデモによって得られたコンテキスト解析の結果について考察する。

図 6.3: ORF の概観

実験環境

展示スペースは、2 m × 2 m の区画が 8 つで、各ブースの足元にセンサデバイス uPart を 19 個設置した。設置方法は、来場された方側からは見えないブース裏で、足元から約 15cm の位置にテープで固定した。uPart のシンクノードとして、USB bridge を一台用意し ACAS を動作させる PC に接続した。uPart が取得可能なデータは、温度、照度、振動の 3 値で、データの送信頻度は 10 秒に設定した。図 6.4 にセンサの設置場所と設置方法を載せた。

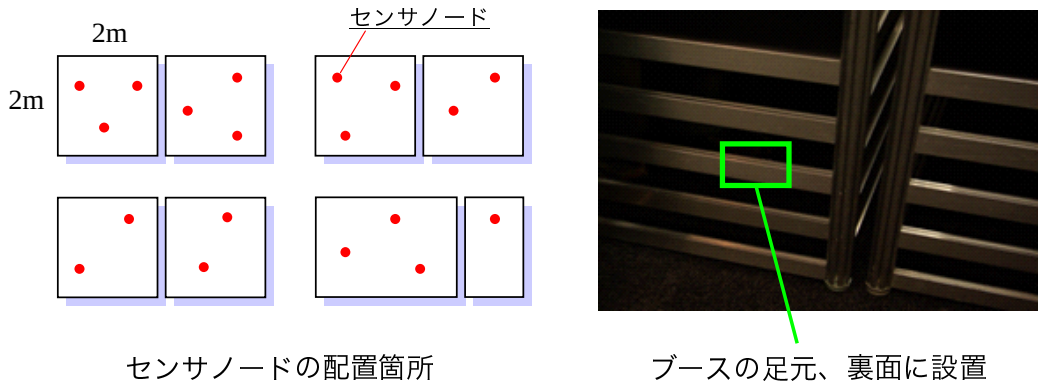


図 6.4: センサ設置箇所の概略

対象となるコンテキスト

この実験では、ACAS のファジィ推論によるコンテキスト解析がどれだけ正しく解析できているかを計測することが目的である。そのため、コンテキストの対象となるものは会場で得られるものに限定され、今回の実験では「賑わい」や「混雑」という状況をコンテキストとして定義した。賑わいや混雑といったコンテキストを uPart の振動センサの値によって判断するために、以下に挙げた条件でプロダクションルールを作成した。表 6.2 に賑わいのプロダクションルールを載せる。

・照度の変化が大きければ、賑わっている
・照度の変化が小さければ、賑わっていない
・現在の振動データが大きければ、賑わっている
・現在の振動データが小さければ、賑わっていない
・一分間の振動データの総計が多いほど、賑わっている
・一分間の振動データの総計が少ないほど、賑わっていない

表 6.2: 「賑わい」のプロダクションルール

if light is big then crowded is big.
if light is small then crowded is small.
if now-movement is big then crowded is big.
if now-movement is small then crowded is small.
if minute-movement is big then crowded is big.
if minute-movement is small then crowded is small.

アプリケーションの動作説明

図 6.5 に実験のため作成したアプリケーションのスクリーンショットを載せた。右端のリストには受信されているセンサノードの ID が表示され、下のリストには受信された生データがリアルタイムに表示される。メインの画面には、ブースの配置図が書かれてあり、色の濃さによってそのブースの賑わいを示している。色が濃くなるほど人が多くいて賑わっていることを意味する。

6.3.2 実験結果

このアプリケーションは、ORF の会期中フルタイムで起動させ続けた。また、それとは別に時間ごとの、おおよその来場者の数を記録として取っていた。その記録を元に、コンテキスト解析結果と実際の賑わいとを比較する。

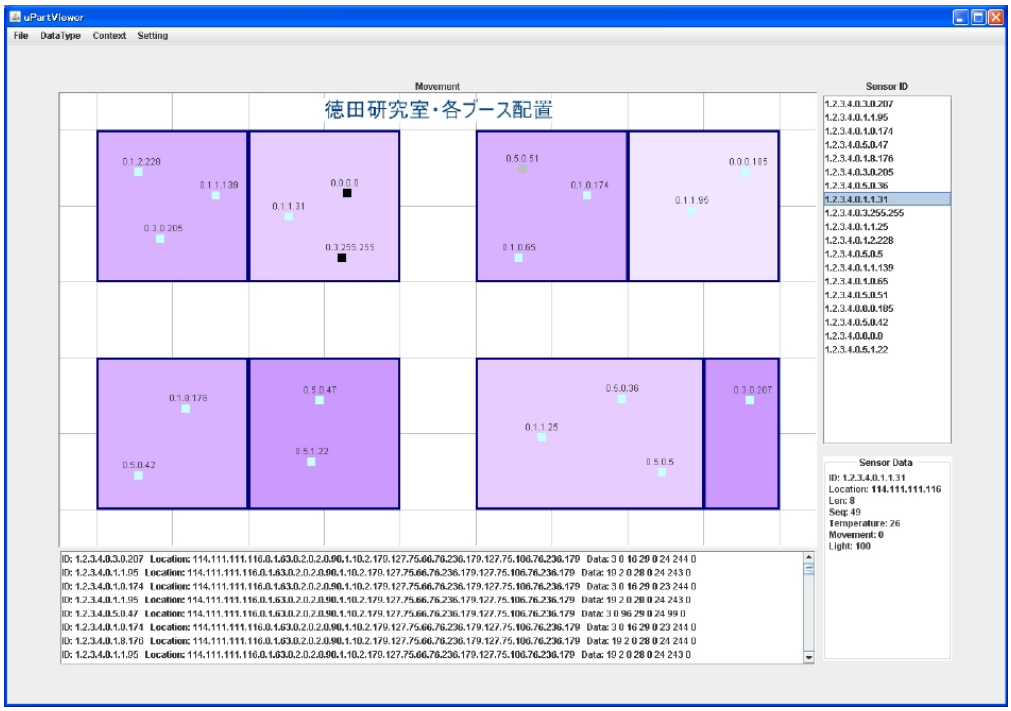


図 6.5: アプリケーションのスクリーンショット

6.3.3 解析結果との比較

解析結果との比較方法は、それぞれの時間帯である程度賑わってきた時の人数を記録しておき、それと同時刻のコンテキスト解析結果を度合いと共に保存する。そして、実際の人数とコンテキスト解析結果とで値に同様の増減が見られるのかを比較した。

表 6.3: 来客者と解析結果の平均値

項目	22 日 13:00	22 日 17:00	23 日 11:30	23 日 14:00	23 日 17:00
来場者数	12 人	11 人	6 人	9 人	5 人
解析結果の度合いの平均	0.4	0.3	0.2	0.2	0.1

ここで、表 6.3 にある解析結果の度合いの平均というのは、アプリケーションは各ブースごとに賑わいの度合いを出力するため、それらの平均値から全体の賑わいを求めた。来場者数は 8 つのブース全ての来場者を合計した人数であるため、このような計算を行っている。図 **??**から実際の来場者数と解析結果の度合いは、関係があるように見えるので、本システムは正しく解析を行えていることが分かる。

6.3.4　実証実験についての考察

以上のように ORF で行った実証実験は、ほぼ予想通りの結果が得られた。しかしながら、この実験に関してはいくつか改善すべき点もあった。それは、センサを使った予備実験を行う必要があったということである。今回の実験では、足元につけたセンサの振動データだけで賑わいというコンテキストを導き出しているのだが、実際は振動センサだけで賑わいを完全に判別することはできなかった。おそらく、音センサや人感センサなどもっと利用できる機器は多くあったはずである。そのようなセンサデータを複合して、ACAS にて解析を行えばより精度の高い解析結果が得られるだろう。また、ブースの骨組みが金属でできていたため、シンクノードから遠い位置に付けたセンサからのデータが非常に届きにくく、そのような環境面でのテスト不足などがあったことも、今後気をつけなければならない問題である。

この実証実験は、実験だけを目標としていたのではなくデモと同時に行っていたため来場者数のカウントなどに僅かながら誤差が生じていると思われる。しかし、このような少しの誤差であれば ACAS による曖昧さの範囲で十分カバーできるため、コンテキスト解析の結果としては信頼に値するものであった。

6.4　本章のまとめ

本章では、ACAS を定量的評価と定性的評価の面から考察した。ACAS を利用することで従来のルールベースに比べて、動的な環境変化に対応できる場合が多いことが分かった。定性的な評価から ACAS では他の手法に比べ、重み付けやコンテキストの

強さといった表現を容易に実現することができる点が強みであることを述べた。また、公共空間において ACAS を用いた実証実験の詳細について論じた。この実験の結果、センサデータの値が頻繁に変わるような状況であっても、ほぼ正確な判別が可能であることを示した。ルールベースなどの従来手法に比べて、ACAS を利用することで有効な場面があることを提示できた。しかしながら、本章での評価ではセンサデバイスが uPart だけという状況であり、異種センサデバイスを用いた評価実験ができなかったことが今後の課題である。

第7章　結論

本章では、本論文にて提案した曖昧さを考慮したコンテキスト解析手法の考察を行い、今後の課題について論じる。そして最後に、本論文全体のまとめを述べる。

7.1　今後の課題

設計とプロトタイプの実装、システム評価から得られた結果について考察し、今後の課題を以下に列挙する。

- **評価シナリオの追加**

  コンテキストアウェアシステムの展開するであろう領域に比べて、本論文の評価で用いたシナリオは非常に限定的なものであった。そのため、ACAS を利用することで効率が上がるかどうかを見極めるには、評価シナリオが少なかったのではないかと思う。今後は、複数のシナリオを追加した上で、継続して評価を取りたいと思う。

- **ファジィ理論の拡張**

  本論文の実装では、外部の Java ライブラリで定義されているファジィ理論の API を利用した。本来ならば、メンバシップ関数の線形をより複雑なものにすることや、ファジィ推論後の重心を求める際のパラメータもユーザが自分で構築できるようなものにしたいと考えていた。そのため、本実装ではファジィ理論を完全に再現したものにならなかった。評価の点から見ても、パラメータがある程度固定的になってしまったため、アルゴリズムの再考が必要であると思う。実際のファジィ制御では、ファジィ理論によるコントロールを何度も繰り返すことでより正確な値へと近づけていく。施行ごとにパラメータを少しずつ最適化することがファジィの特性でもあるので、その拡張も可能である必要がある。

- **システム全体の実用性**

  実装に関して、上記のファジィ理論を拡張する部分が不完全である点だけでなく、ACAS のシステム全体を見たときにも改善すべき所が多い。まず、センサデータの形式を外部から取り込んで、そのセンサをシームレスに利用できるようにす

るモジュールが未実装である。そして、コンテキスト解析を行った後にアクチュエータに対して命令を出す部分について、システム構成では ACAS API によってプロキシコントローラに命令を代替させる構想でいたのだが、本論文ではそこまで実装することができなかった。今後はそれらを実装することで、ACAS がよりシステムとして実用的になると考えられる。

7.2 ACAS の展望

本論文で提案した ACAS について、どのような展開が考えられるのかを述べ、今後の展望について論じる。

解析精度の向上

ファジィ理論による推論を機器制御に利用する場合、センサデータだけでなく複数の外的な要因も合わせて解析することが望ましい。例えば、ウェブ上にあるポータルサービスから情報を引き出して、その中からユーザの必要となる情報をコンテキストの条件として内包させることなど、情報を複合的に利用することが考えられる。本論文では、そのような外的要因を想定環境を設定することで排除していた。そのためファジィ推論のプロダクションルールは、センサデータに関する記述だけに留まり限定的であったと言える。ルールベースではルールの構築が複雑すぎるような条件も、人間に分かりやすい形で記述できることがファジィ理論の利点でもあるので、センサデータだけでなく、さまざまな情報を入力値として記述できるように拡張する必要がある。さまざまな形式の情報をコンテキスト解析に含めることで、コンテキストの定義を柔軟にすることができ、多面的に見ることが可能となる。引いてはそれが、コンテキスト解析の精度を向上させることにつながる。

ACAS の利用範囲の拡大

本論文では、ACAS を利用することで、ルールベースによるコンテキスト解析手法よりもセンサデータを柔軟に扱うことが可能であることを示した。この柔軟さは、解析にファジィ理論を利用したことによる意味が大きい。従来ファジィ理論は工業分野において大規模な機器の制御などや、家電製品の制御基板内に組み込まれることが多く、センサネットワークにおけるコンテキスト解析にはそれほど注目されていなかった。しかし、ACAS プロトタイプの評価からコンテキストアウェアシステムに組み込むことで、十分な利点を得られることが分かった。二値理論とは異なり、ファジィ理論による曖昧さは、コンテキスト解析のように多様性を含んだ事象には特に効果的で

ある。コンテキストアウェアシステムの普及によって、コンテキスト解析に求められる機能が多岐にわたることが考えられる。そのような時、さまざまなコンテキストに柔軟に対応することのできる ACAS は、新たな手法として利用することができるのではないだろうか。

7.3 本論文のまとめ

本論文では、曖昧さを考慮したコンテキスト解析手法である ACAS の提案とその評価を行った。まず、既存のルールベースを用いたコンテキスト解析手法の問題点について考察し、どのような解決策が考えられるのかをまとめた。そして、問題の解決策としてコンテキスト解析に曖昧さを導入する手法 Ambiguity-based Context Analysis System (ACAS) の提案を行った。センサデータそのものに含まれる誤差や、環境の動的な変化などのコンテキスト取得の際の外的な要因によって、正確なコンテキスト解析ができないという問題に対して、ACAS ではファジィ理論による曖昧さを利用することで、システムに柔軟性を持たせたコンテキスト解析を可能にした。柔軟性を持つことによって、確率によるコンテキストの把握よりも、人間の思考プロセスに近い判断を下せるため、より現実的なコンテキスト解析ができる。コンテキストの表現は、プロダクションルールの定義によって行われるため自然言語による表現が可能となり、従来手法の固定的な閾値による設定とは大きく異なっている点が特徴である。

ACAS のプロトタイプによる評価では、想定する環境においてルールベース手法と ACAS による解析とで同じコンテキストを定義し、その解析結果の比較を行った。その結果、環境の動的な変化では ACAS は十分に有効であるということを証明できた。また、公共空間における実証実験の結果からも、ファジィ理論をコンテキスト解析に利用することは十分に可能であり実用に耐えうるものであることを示せた。

今後は、実装が不十分であった部分の手直しをする必要がある。前述の ACAS の展望の節で述べたとおり、解析精度の向上や ACAS の利用範囲を拡大させることが急務である。そして、ACAS を手法としての提案だけに留まらせるのではなく、API を整えて、人に使ってもらえるシステムにすることが本研究の最終的な目標となる。

謝辞

本研究を進めるにあたり、御指導を頂きました慶應義塾大学環境情報学部教授 徳田英幸博士、慶應義塾大学環境情報学部准教授 高汐一紀博士に深く感謝致します。

慶應義塾大学 徳田・村井・楠本・中村・高汐・重近・湧川・バンミーター合同研究室の諸先輩方には、折に触れ暖かいご指導と御助言を頂きましたこと拝謝いたします。論文執筆にあたって特別な御助言を頂きました、間 博人氏、斉藤 匡人氏、高橋 ひとみ氏、森 雅智氏をはじめとする慶應義塾大学徳田研究室 ECN グループの諸氏に深く感謝しております。また、ECN グループの OB である青柳 禎矩氏、金田 裕剛氏には在学中に多くのご指導をいただきました、ここに深い感謝の意を表します。

最後に、何不自由のない大学生活を送れるように支えてくれた両親とたくさんの友人に感謝し、本論文の謝辞と致します。

2008 年 1 月 31 日

藤本 俊佑

参照論文

- 藤本 俊佑, 金田 裕剛, 斉藤 匡人, 高汐 一紀, 徳田 英幸.
  “ContextSwitch: センサネットワークにおけるレイヤ化コンテキスト切り替え機構” 第 5 回 SPA スプリングワークショップ, ポスター発表
  2007 年 3 月.

参考文献

[1] Crossbow Technology, Inc. http://www.xbow.com/.

[2] Edward Sazonov: Home Page. http://people.clarkson.edu/ esazonov/FuzzyEngine.htm.

[3] Michael Hinz, Stefan Pietschmann, and Zoltan Fiala. A Framework for Context Modeling in Adaptive Web Applications. In *IADIS International Conference WWW/Internet06*, 2006.

[4] Kenji Hisazumi, Tsuneo Nakanishi, Teruaki Kitasuka, and Akira Fukuda. CAMPUS: A Context-Aware Middleware. In *The 2nd CREST Workshop*.

[5] Markus C. Huebscher and Julie A. McCann. An adaptive middleware framework for context-aware applications. In *Pers Ubiquit Comput 10*, pp. 12–20, 2006.

[6] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An Operating System for Sensor Networks.

[7] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TinyDB: An Acquisitional Query Processing System for Sensor Networks. In *ACM Transactions on Database Systems*, 2005.

[8] Samuel R. Madden, Michael J. Franklin, and Wei Hong. TinyDB: In-Network Query Processing in TinyOS. 2002.

[9] Anand Ranganathan, Jalal Al-Muhtadi, and Roy H. Campbell. Reasoning about Uncertain Contexts in Pervasive Computing Environments. In *IEEE Pervasive Computing*, pp. 62–70, 2004.

[10] Bill N. Schilit, Norman Adams, and Roy Want. Context-Aware Computing Applications. In *In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, pp. 85–90, 1994.

[11] Smart Space Laboratory. http://www.ht.sfc.keio.ac.jp/SSLab/.

[12] SoftComputing lab. http://scl.m-kb.net/.

[13] Universitat Karlsruhe. TecO : http://www.teco.edu/.

[14] uPlatea. http://www.ht.sfc.keio.ac.jp/uPlatea/index.htm.

[15] Mark Weiser. The Computer for the Twenty-First Century. In *Scientific American*, 1991.

[16] L. A. Zadeh. Fuzzy Sets. In *Information and Control*, pp. 338–353, 1965.

[17] クロスボー株式会社. http://www.xbow.jp/motemica.html.

[18] 永原 崇範, 鹿島 拓也, 猿渡 俊介, 川原 圭博, 南 正輝, 森川 博之, 青山 友紀, 篠田 庄司. ユビキタス環境に向けたセンサネットワークアプリケーション構築支援のための開発用モジュール U3（U-cube）の設計と実装. 電子情報通信学会, 2003.

[19] 久住 憲嗣, 中西 恒夫, 北須賀 輝明, 福田 晃. ファジィ理論を応用したコンテキストのクオリティ表現. In *DICOMO2003*, 2003.

[20] 金 勝進，司 化，川西 直，森川 博之. 認知症患者の日常生活動作を支援するコンテキストアウェア案内システムの設計と実装. 情報処理学会ユビキタスコンピューティングシステム研究会 (UBI), 2007.

[21] 寺野 寿郎, 浅居 喜代治, 菅野 道夫. 「応用ファジィシステム入門」. 朝倉書店, 1989.

[22] 小林 英嗣, 依田 育生. 映像を利用した直感的な操作によるコンテキストアウェアサービスの制御条件設定方法. 情報処理学会研究報告「ユビキタスコンピューティングシステム」, 2006.

[23] 小林 英嗣, 小河原 成哲, 依田 育生. ホームネットワークにおける複数サービスの統括的制御システム. 電子情報通信学会技術研究報告, 2004.

[24] 上田 真由美, 梶田 将司, 間瀬 健二. 学習操作履歴を用いた利用者コンテキストアウェアな CMS ツールに関する検討.

[25] 西垣 弘二, 安本 慶一, 柴田 直樹, 伊藤 実. コンテキストに基づいた情報家電の連携を実現するためのフレームワークおよびルールベース言語の提案. 情報処理学会ユビキタスコンピューティングシステム研究会 (UBI), 2004.

[26] 青山・森川研究室. http://www.mlab.t.u-tokyo.ac.jp/.

[27] 藤波 香織, 山邉 哲生, 中島 達夫. コンテクストアウェアなアプリケーションフレームワークにおけるメタコンテクスト情報の利用方法の提案とその応用.

[28] 徳田・高汐研究室. http://www.ht.sfc.keio.ac.jp/.

[29] 南 正輝, 森川 博之, 青山 友紀. ネットワークサービスシンセサイザのデザイン. 電子情報通信学会技術研究報告, pp. 1–8, 2001.

[30] 日本ファジィ学会編. 「ソフトコンピューティング用語集」. オーム社, 1996.

付録

FuzzyEnginのサンプルコード

ファジィ理論を実装したライブラリであるFuzzyEnginのサンプルコードを載せる。各メソッドについて適宜コメントを入れた。

```
 1: /**
 2:  * FuzzyEngin サンプルコード
 3:  *
 4:  * [シナリオ]
 5:  * レストランで食事の際に、サービスの質 (service) と料理のおいしさ (food) から
 6:  * チップの金額を決める。食事代に上乗せする分の 0〜20(%) が出力される。
 7:  *
 8:  */
 9: public class FuzzyEngine {
10:   public static void main(String[] args) {
11:     // 出力結果を入れる変数
12:     double result = 0.0;
13:
14:     // service のパラメータについてメンバシップ関数を作成
15:     LinguisticVariable service = new LinguisticVariable("service");
16:     service.add("poor", 0, 0, 2, 5);
17:     service.add("soso", 2, 4, 6, 8);
18:     service.add("good", 5, 8, 10, 10);
19:
20:     // food のパラメータについてメンバシップ関数を作成
21:     LinguisticVariable food = new LinguisticVariable("food");
22:     food.add("not delicious", 0, 5, 5, 10);
23:     food.add("soso", 5, 10, 10, 15);
24:     food.add("delicious", 10, 15, 15, 20);
25:
26:     // tip のパラメータについてメンバシップ関数を作成
27:     LinguisticVariable tip = new LinguisticVariable("tip");
28:     tip.add("small", 0, 5, 5, 10);
29:     tip.add("average", 5, 10, 10, 15);
30:     tip.add("big", 10, 15, 15, 20);
31:
```

```
32:     // FuzzyEngine を作成
33:     FuzzyEngine fuzzyEngine = new FuzzyEngine();
34:
35:     // FuzzyEngine にパラメータの登録
36:     fuzzyEngine.register(service);
37:     fuzzyEngine.register(food);
38:     fuzzyEngine.register(tip);
39:
40:     try {
41:       // service と food の値を入力する
42:       service.setInputValue(3);
43:       food.setInputValue(3);
44:
45:       // ファジィ理論における解析基準を定義する
46:       fuzzyEngine.evaluateRule
47:         ("if service is poor and food is not delicious then tip is small");
48:       fuzzyEngine.evaluateRule
49:         ("if service is poor and food is delicious then tip is average");
50:       fuzzyEngine.evaluateRule
51:         ("if service is soso and food is not delicious then tip is small");
52:       fuzzyEngine.evaluateRule
53:         ("if service is soso and food is soso then tip is average");
54:       fuzzyEngine.evaluateRule
55:         ("if service is soso and food is delicious then tip is big");
56:       fuzzyEngine.evaluateRule
57:         ("if service is good and food is not delicious then tip is small");
58:       fuzzyEngine.evaluateRule
59:         ("if service is good and food is soso then tip is average");
60:       fuzzyEngine.evaluateRule
61:         ("if service is good and food is delicious then tip is big");
62:
63:       // 各パラメータを合成してその重心を求める
64:       // 重心を非ファジィ化することで重み付けされた値が出力される
65:       // result には 0〜20 の値が格納される
66:       result = tip.defuzzify();
67:
68:     } catch (Exception e) {
69:       e.printStackTrace();
70:     }
71:     System.out.println("result = " + result);
72:   }
73: }
74:
75: [EOF]
```