

PRINTEPS におけるロボット喫茶店の来店客対応と席案内の実現と評価 Implementation and Evaluation of visiting service and the seat guide in Robot Café using PRINTEPS

菅原 優^{*1} 丸川 大輝^{*1} 番原 常公^{*2} 田中 康浩^{*2} 森田 武史^{*2} 斎藤 英雄^{*2} 山口 高平^{*2}
Yu Sugawara Daiki Marukawa Hisatomo Banbara Yasuhiro Tanaka Takeshi Morita Hideo Saito Takahira Yamaguti

^{*1} 慶應義塾大学大学院理工学研究科
Keio University, Graduate School of Science and Technology

^{*2} 慶應義塾大学理工学部
Keio University, Faculty of Science and Technology

We are developing PRACTICAL INTELIGENT aPPlicationS (PRINTEPS), which is a platform for developing comprehensive intelligence applications only by combining four types of modules such as knowledge reasoning, speech dialog, image sensing and manipulation. In this paper, we refer to the implementation and Evaluation of visiting service and the seat guide in Robot Café using PRINTEPS.

1. はじめに

現在、知識推論、音声対話理解、画像センシング、マニピュレーションなどに関連するソフトウェアモジュールを再構築するだけで、人と機械が協働可能な総合知能アプリケーションを開発するプラットフォーム PRINTEPS (PRACTICAL INTELIGENT aPPlicationS) の研究を進めている[山口 15]。本稿では、PRINTEPS における喫茶店業務のうち、画像センシングと知識処理の統合により、喫茶店業務の来店客対応と席案内の実現と評価を行う。

2. ルールセットと画像センシングの統合

人の業務をロボットが代行する場合、人間ができることとロボットができることは異なるため、人間の業務ルールをロボット向けの業務ルールに修正する必要がある。人間の世界では、豊富に存在する業務ルールを活用し、かつルールの修正に柔軟に対応する枠組みとして BRMS (Business Rule Management System) [森田 14] が大きな注目を浴びるなど、業務ルールを活用したシステム開発の流れが活発になっている。

しかし、業務ルールを直接ロボットが活用するのは大変困難である。なぜなら、人とロボットで知覚できるものが異なるためである。人とロボットの知覚の粒度の違いを図 1 に示す。よって、ロボットの知覚に当たる画像センシングを豊富な業務ルールセットに対してルールの修正に柔軟な形で結びつける枠組みが必要とされる。

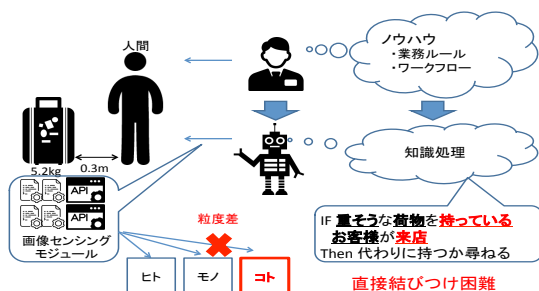


図 1 人とロボットの知覚の粒度の違い

連絡先: 丸川大輝, 慶應義塾大学大学院理工学研究科
〒223-8522 神奈川県横浜市港北区日吉 3-14-1
TEL: 045-566-1614
daiki.marukawa@keio.jp

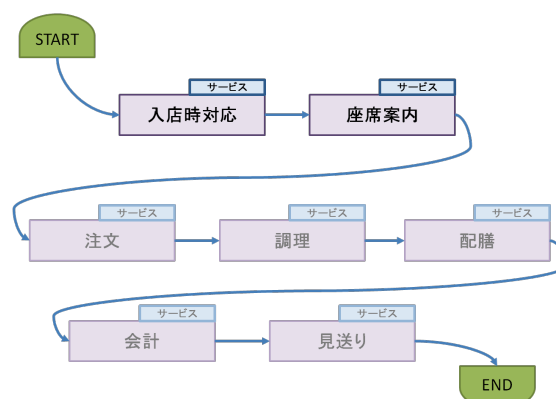


図 2 ロボットカフェのワークフロー

つまり、ルールセットを実世界にマッピングするには、画像センシングから得られた物理情報がある記号に変換する他に、さらにそれらをルールセットに結びつける枠組みが必要である。

そこで、本稿では言葉で定義されたサービスルール群を実世界にマッピングすることを目指し、ルールに重きを置き、ルールセットと信号を繋ぐことを考える。なお、本稿では画像センシング自体には焦点を当てず、詳細については割愛する。

その実現方法としてストリーム推論[市瀬 15]の一種である、C-SPARQL (Continuous SPARQL) [Davide 10] を用いてルールと画像センシングを統合する手法を提案する。C-SPARQL とは RDF (Resource Description Framework) トリプルにタイムスタンプをつけた RDF ストリームに対し、SPARQL を拡張した検索言語を使用することでイベント処理が可能になるというものである。これにより、業務担当者は画像センシングの結果を直接意識するのではなく、C-SPARQL が検出するイベントを通して考えることで、業務担当者が持つルールセットをロボットが利用可能になり、より豊かなロボットサービスを構築することができる。

3. システム概要

本稿では、PRINTEPS を喫茶店業務で活用する。喫茶店業務のワークフローを図 2 に示す。本稿では入店時対応と席案内に焦点を当てる。システム構成図を図 3 に示す。

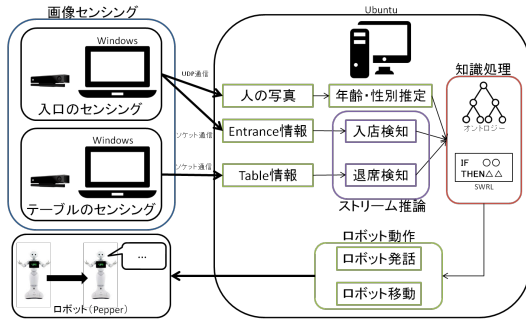


図 3 システム構成図

```
REGISTER QUERY CustomerDetectionQuery AS
PREFIX f: <http://larkc.eu/csparql/sparql/jena/ext#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX printeps: <http://printeps.org/cafe/entrance/>

SELECT ?s (COUNT(?s) AS ?cnt) (AVG(?distance1) AS ?distance)
FROM STREAM <http://printeps.org/cafe/entrance> [RANGE 3s STEP 1s]
WHERE {
  ?s rdf:type printeps:Customer.
  ?s printeps:positionAtTime ?ts1;
    printeps:positionAtTime ?ts2.
  ?ts1 printeps:distance ?distance1.
  ?ts2 printeps:distance ?distance2.
  BIND(?distance2 - ?distance1 AS ?difference)
  FILTER((f:timestamp(?s, printeps:positionAtTime,?ts1)
    > f:timestamp(?s, printeps:positionAtTime,?ts2)) && 0.1 < ?difference)
}
GROUP BY ?s HAVING (AVG(?distance1) < ?length && 1 < COUNT(?s))
```

図 4 入店を検知する C-SPARQL クエリ

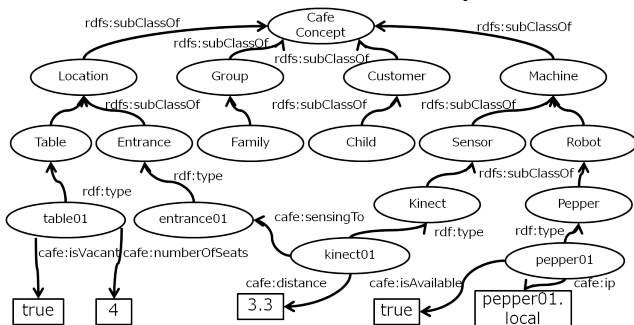


図 5 喫茶店オントロジーの一部

入口のセンシング結果やテーブルのセンシング結果が Windows マシンによって得られ、それらの結果を UDP 通信またはソケット通信を用いて Ubuntu マシンに送信される。そして送信された情報を基に、ストリーム推論にかけられて入店または退席というイベントを検知するか、年齢・性別判定にかけられる。それらから得られたイベントないしは属性情報を知識処理に用い、知識処理から得られた結果を基にロボット動作のプログラムが実行される。このプログラムからの指令に基づきロボットが動作・発話を行う。これら一連の手順を踏まえることで、ロボットカフェの入店時対応と座席案内というサービスを実現している。

それぞれの構成要素においてどのようなシステムを構築したのかを詳細に説明していく。

3.1 C-SPARQL を用いたストリーム推論

ストリーム推論部として、C-SPARQL を用いたシステムを構築した。この部分では、画像センシングから受け取った処理結果をリアルタイムに受け取り、それらを一時的に蓄積したものを利用して、C-SPARQL クエリに基づいてイベント検知を行い、イベントが検知された時に知識処理以降の処理を行うという形になっている。例として、入店時挨拶において入店を検知する C-SPARQL クエリを図 4 に示す。このクエリでは、直近 3 秒間に

```
Entrance(?entrance), GreetingAtEntrance(?service),
Group(?group), Pepper(?pepper), hasGroup(?entrance, ?group),
robotPosition(?pepper, ?entrance), servedBy(?service, ?pepper),
servedTo(?service, ?group), numberOfCustomers(?group, 2)
-> greeting(?pepper, "いらっしゃいませ。2名様ですか？")
```

図 6 入店時挨拶を決定するルール例

おけるデータに対して検索を行っており、画像センシングが認識したある人物の距離と、時間経過による移動の向きに注目している。具体的には、認識された人物とセンサーの距離の平均値から店内であるか推論し、タイムスタンプを基に移動方向が店内方向かを推論している。

3.2 オントロジー

ロボットカフェ実現のために、喫茶店オントロジーを構築した。喫茶店オントロジーの一部を図 5 に示す。喫茶店オントロジーは、のちに説明する SWRL ルールで利用する情報や、C-SPARQL 内で入口からセンサーまでの距離として利用される情報など、他の知識ベースで利用される情報を定義している。また、座席数がいくつあり、空いているか埋まっているかなど喫茶店内の状況も喫茶店オントロジーに反映されており、喫茶店オントロジーにより喫茶店内の情報を管理している。

3.3 SWRL を用いたルール

本研究では、全校で説明したオントロジーのクラスやインスタンスネットワークを利用して、その状況に合わせた動作、発話を行うため、SWRL ルールを用いた。業務ルールとして、年齢からお客様のクラスを決定するルールや、グループの構成からグループのクラスを決定するルール、入店時挨拶の内容を決定するルールなどを定義している。

図 6 に、入店時挨拶の内容を決定するルールの一つを示す。これは、お客様のグループやロボットが入口にいる際に、グループの人数が 2 名だったら、そのロボットが「いらっしゃいませ。2名様ですか？」と発話するルールとなっている。このように、ルールによって発話内容を定義することで、ルールの変更のみでロボットの挙動を変化させることができる。

3.4 入店時挨拶のワークフロー

入店時対応サービスは、お客様に挨拶、満席時の場合待つことが可能か確認、満席の場合待ち行列に並ぶことを指示、待ち時間が長そうな場合メニューを配るといった四つのプロセスから構成されている。図 7 にワークフローを示す。

お客様に挨拶プロセス内で C-SPARQL を用いてお客様の入店というイベントを検知し、そのお客様の人数を基にお客様に「いらっしゃいませ。〇名様ですか？」のような挨拶を行う。その後、カフェオントロジーから取得される店内の状況に応じてロボットが発話や動作を行うというシナリオとなっている。このうち、C-SPARQL モジュールを利用しているお客様に挨拶プロセスの詳細なワークフローを図 8 に示す。

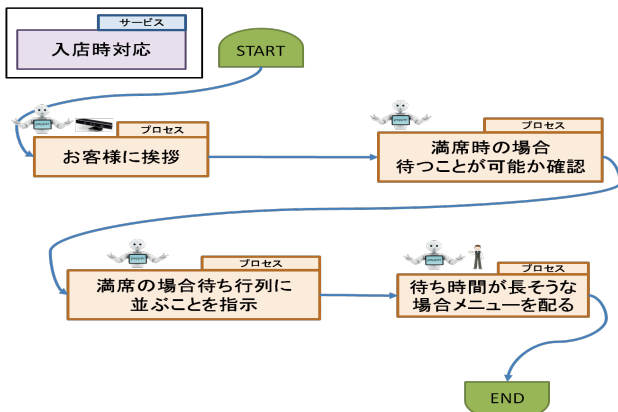


図 7 入店時対応サービスのワークフロー

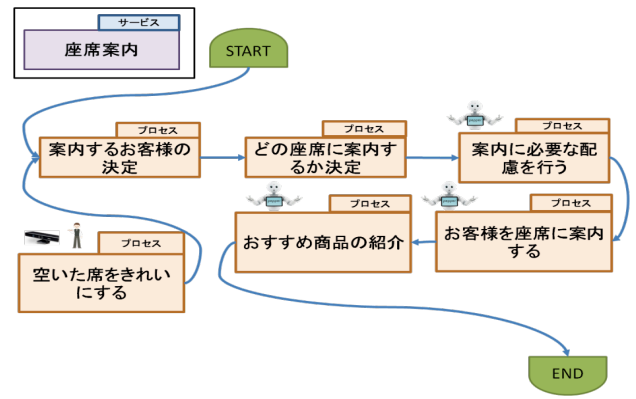


図 9 座席案内サービスのワークフロー

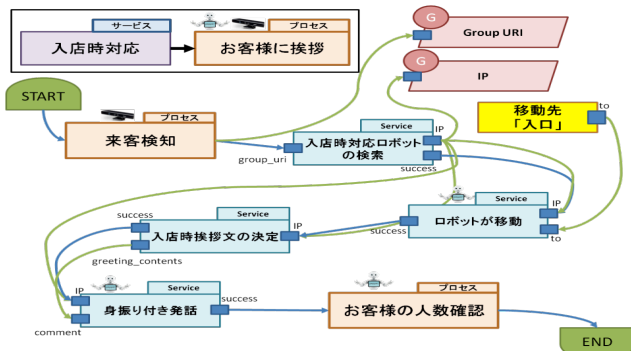


図 8 お客様に挨拶プロセスのワークフロー

お客様に挨拶プロセスの流れとしては、まず来店検知サブプロセス内で、入口にある Kinect がセンシングした結果を受け取り、RDF ストリームを生成する。そのストリームを用いて、C-SPARQL モジュールが来店検知を行い、来店が検知されたときにその来店客の人数を登録するというのが、来店検知サブプロセス内での主な処理の流れである。また、来店検知に合わせて、検出した人の顔を検出し、そのデータも Kinect センサー側から受信している。この写真を利用して、別の処理プロセス内で、年齢認識・性別認識のサーバーにその画像を送信、その結果を受け取ってオントロジー上のお客様情報を更新することで、のちのサービス内で年齢や性別に応じた処理を可能としている。その後、その来店グループに対応するロボットを、カフェオントロジーのロボット使用状況に基づき選択し、そのロボットが環境地図を利用した経路計画結果に基づき移動、その後入店時挨拶の内容を決定したのちにその発話内容を発話する。このとき、入店時挨拶内容は greeting プロパティによって決定されているため、発話時は SWRL ルールによって greeting プロパティにひもづいた動作を実行しながら発話を行う。その後、お客様の返答に応じて必要があれば人数の修正をお客様の人数確認サブプロセス内で実行する。このプロセス内で取得したお客様情報と、オントロジー内で蓄えられている空席、満席などの店内情報を基に、その後のプロセスも実行されており、そのときのカフェの混雑状況によってそれ以降のプロセスがどのように実行されるかが動的に変化している。

3.5 座席案内のワークフロー

座席案内サービスのワークフローを図 9 に示す。お客様を案内できる状態であれば、入店時対応サービスから座席案内サービスが呼び出されたときに、直接そのお客様に対しての座席

案内が始まるようになっている。その際には、どの座席に案内すべきかを選択し、その後お年寄り用のメニューやお子様用の椅子など、案内に必要な配慮があるようだったらその配慮が必要かどうかを確認し、その後実際にお客様を案内し、お客様が席についたのちにそのお客様グループの分類されたクラスに基づいた商品のおすすめを行うといった流れになっている。

入店時対応サービスを行った際に、満席であったらお客様に待ってもらうということになる。その際には、座席案内サービスの空いた席をきれいにするプロセス内で C-SPARQL により退席というイベントの検知を行う。そして、退席を検知した際には、空いた席をきれいにするプロセスが実行され、それにより空席を準備し、空席が準備されたところで案内するお客様の決定プロセス以降が実行される仕組みとなっている。

空いた席をきれいにするプロセスは、サブプロセスである座席が空いたことを検知プロセス、オーナーによる机の片付けプロセス、机の上が片付いたのを確認プロセスの三つからなる。このプロセスでは、座席が空いたことを検知プロセス内で C-SPARQL による退席検知を行い、退席イベントが検知された場合にはオーナーによる机の片付けプロセスでコンソールを介して喫茶店オーナーに退席が検知されたテーブル名と座席の準備を要求する旨を表示する。そして、机の上が片付いたことを確認プロセスでは、机の上の片付けが終わったら喫茶店オーナーにそのことを入力してもらい、それによって空席が生まれたと確定させ、オントロジーの更新を行う。そして、それ以降の処理が行われ、Pepper による座席案内へと連鎖していく仕組みになっている。

4. 実験

本手法で構築されたシステムの修正容易性を評価するために、実験を行った。構築したロボットサービスをお客様に体験してもらい、その声を集め、分析することで、喫茶店マスターの気づきに当たる改善策を擬似的に得て、それに対する修正コストを考えることとする。今回の実験は以下のような状況で行った。

- ・場所: 慶應義塾大学矢上キャンパス 07-304
- ・対象: 当研究室所属の学生 6 名

対象の学生 6 名をロボットカフェにおけるお客様と仮定し、満席時における入店時対応サービスと座席案内サービスを体験してもらい、各被験者にアンケートに答えてもらった。アンケート結果からサービスの質が向上すると考えられるものについて、改善策としてあげられるものを列挙し、その修正に必要な作業のコストをまとめたものを表 1 に示す。なお表 1 におけるコストで記載した記号とそれぞれの意味の対応表は表 2 に示す。

表 1 ロボットカフェの改善案と作業コスト

番号	改善案	コスト
1	ロボットが音声認識をするタイミングを伝える	○
2	どのタイミングで、どのように返事すべきか発話	○
3	より多くの動作を組み込む	○
4	待つ場所をしっかりと伝える	○
5	来客を検知したら、動き出す前に認識したことを発話する	□
6	待ち時間に何かロボットからはなしかける	△
7	移動しながら発話を行えるように	△
8	現在の満席情報などをタブレットに表示	△
9	荷物を認識して、大きい荷物をもっていったら座席案内の際に配慮する	☆
10	人の顔の位置を交互に見ながら発話する	★
11	検出した人の方向を指さしながら発話する	★
12	待ち時間をお知らせする機能を作る	x
13	多言語で対応出来るように	x
14	待機中にPepperアプリで遊べるように	x
15	移動中に手を動かしてコミカルに	x

表 2 コスト記号と作業内容の対応

コスト記号	作業内容
○	SWRLルール、オントロジーなどの変更
□	既存モジュールの再利用、実行順序の変更
△	モジュール自体の内容を修正、新規モジュール開発
☆	画像処理モジュールがあればSWRLルール、オントロジーの変更で対応可能
★	画像処理モジュールがあればモジュール自体の内容の修正で対応可能
x	ハードウェア制約、対象範囲などで不可能

5. 評価と考察

5.1 画像センシングと知識処理の統合に関する考察

来客検知については、その地点でのセンサーの値である距離のみならず、前後のデータとの差異からどの方向に移動しているかといったことまで考慮したイベント検知を C-SPARQL クエリのみで定義でき、それをを用いることでイベントを検知することが出来た。

また、退席検知については、センサーから得られたテーブルに着席している人数が 0 になったという一時的なセンサーデータの結果のみを用いるのではなく、人数の変化から退席というイベントを定義し、それをを用いてイベントを検知することが出来た。

これらの C-SPARQL というイベント検知の仕組みを用いて画像センシングと知識処理の統合を行ったことで、直接画像センシングと知識処理を統合した際と比較して、以下の三つのメリットがあると考えられる。

- ・時間変化によるセンサーの値の変化に基づいた検知を容易な形で実現できる
- ・画像センシングを分離することによる多くのセンサーとの連携可能性がある
- ・知識処理が分離されることにより、ルールの記述が人間の思考に近い形で記述可能

5.2 修正容易性に関する考察

本項では、前述した実験の結果を基に、修正容易性の考察を行う。

まず、表 2 のコスト記号と本手法のコスト低減への貢献について考察を行う。○は、SWRL ルールや、オントロジーといった宣言的知識の変更によって対応可能なものである。これは、本手法を導入したことで画像センシングと知識処理を独立させたことにより、修正容易な宣言的知識の変更のみで対応出来るので、これは本手法の貢献により修正容易性が向上したと言える。同様に、☆は画像処理モジュールが拡充すればという条件つきではあるものの、プログラムの修正自体は宣言的知識の変更のみで対応可能である。加えて、□は既存モジュールの組み替えで可能であり、プログラムの修正ではあるものの、処理自体を大きく変えるのではなく、実行ファイルの一部の変更のみで対応可能である。これら三つは本手法によって修正容易性が向上した箇所と言えるであろう。

以上のことから、表 1 に示した改善案のうち、本手法によって修正容易性が向上した数は、6 個である。今回の評価の対象外

である×の数を除いた改善案数が 11 個であるため、修正容易性が向上した割合は $6/11 \approx 54.5\%$ であり、半分以上の改善案について本手法によって修正容易性が向上したと言える。つまり、本手法を用いることで通常のロボットアプリケーション開発と比較して修正容易性が高まったということがこの実験結果から言える。

システムの初期構築には ROS サーバー側(ロボットの動作や知識処理など自体の処理を記述したもの)の実装を除いた、ROS クライアント側(それぞれのクライアント間の値の受け渡しや変数の定義などを行う)の実装で一週間ほどかかったものの、改善案への対応はわずか二時間ほどで完了した。しかも、その二時間のうちの大半は、動作を選択するのにかかった時間で、具体的には、それぞれの動作を実際に発話と合わせて実行してみて、じっくりくるものを選ぶ時間にかかった。もちろん、通常のアプリケーションとして開発されたものでも、初期開発と比較すると多少の改善はみられるが、全てプログラムベースの修正になってしまうので、修正自体は難しくはなかくとも時間はかかってしまうであろう。初期構築と修正にここまで大きな時間差がかかったのは、本手法の修正容易性によるものであると考える。なお、現在 PRINTEPS プロジェクトで開発中のワークフローエディタを利用すればより短時間でこの修正を行えたと考えられるため、ワークフローエディタの完成が望まれる。また、SWRL ルールについても同様のエディタの開発が行われれば、喫茶店マスターのような、プログラム未経験の業務担当者であってもロボットアプリケーションを修正することが可能になると考えられる。

6. おわりに

本稿では、PRINTEPS の応用として、ストリーム推論を用いてイベントを検知し、画像センシングと知識処理を提案し、ケーススタディとして、来店客対応と席案内を行い、この枠組みを示した。この枠組みにより、ルールセットの変更によってロボットの振る舞いを変化させることが可能となった。

今後の課題としては、より多くの画像センシングにストリーム推論を用いることで、より多くの情報を利用できるようにし、よりサービスを豊かにすることや、実際にサービスを変更する際にどの程度の工数がかかるかといった修正容易性の検証などが挙げられる。

謝辞

本研究は、科学技術振興機構(JST)戦略的創造研究推進事業(CREST)「実践知能アプリケーション構築フレームワーク PRINTEPS の開発と社会実践」の支援によって実施した。

参考文献

- [山口 15] 山口高平, 中野有紀子, 斎藤英雄, 森田武史, 青木義満, 萩原将文, 斎藤俊太, 知能共進化のための実践知能アプリケーションプラットフォーム PRINTEPS, 人工知能学会全国大会(第 29 回)論文集, 114-2 2015.
- [森田 14] 森田武史, 山口高平, 業務ルール管理システム BRMS の現状と動向, 人工知能, Vol. 29, No.3, pp.277-285, 2014
- [市瀬 15] 市瀬龍太郎, ストリーム推論, 人工知能, Vol. 30, No.5, pp.574-579, 2015
- [Davide 10] Davide Francesco Barbieri, Daniel Braga, Stefano Ceri, Emanuele Della Valle, Michael Grossniklaus, Querying RDFstreams with C-SPARQL, SIGMOD Record 39(1), pp.20-26, 2010