ライフログサービスのためのオントロジに基づく行動イベント処理

中村　亮† 石川　佳治† 杉浦　健人† 脇田佑希子†

† 名古屋大学大学院情報科学研究科　〒 464–8603 愛知県名古屋市千種区不老町

E-mail: †{rnakamura,sugiura}@db.ss.is.nagoya-u.ac.jp, ††ishikawa@is.nagoya-u.ac.jp, †††wackie@nagoya-u.jp

あらまし　近年，モバイル端末の普及やストレージ技術の発展により，大量のデータが生成・蓄積されるようになっている．中でも人間の行動に関するデータに注目が集まっており，センサ等から得られるデータを分析し，行動情報を抽出する研究などが行われている．本研究では，オントロジを用いることにより，センサデータだけでなく複数のデータソースから得られる行動情報を組み合わせ，より高次な行動情報を抽出する手法を提案する．具体的には，行動オントロジの設計と，応用例としてライフログアプリケーションを想定した行動イベント処理システムの構成について述べる．

キーワード　オントロジ，イベント処理，ライフログサービス，セマンティック Web

1. はじめに

近年，モバイル端末の普及やストレージ技術の発展に伴い，大量のデータが生成，蓄積されるようになっている．中でも，データから人間の行動情報を抽出し活用する試みが数多く行われており，そのための技術としてセンサデータを用いて行動認識を行う研究が盛んである．例えば上松らの研究 [1] では，歩行者が持つ携帯端末に取り付けられたセンサを用いて，歩行者の歩幅や歩数を高い精度で推定することに成功している．また，大内らの研究 [2] では，スマートフォンに搭載された加速度センサおよびマイクを用いて，機械学習に基づく手法により，皿洗いなどの室内の生活行動を高精度で認識している．ほかにも行動認識に関する研究が数多く行われており [3, 4]，スマートフォンなどによる行動認識の実現が現実味を帯びている．

このような行動認識技術を様々なサービスやアプリケーションへ応用することが期待されており，ライフログアプリケーションはその一つである．例えば，食事や運動などの行動を認識してデータを蓄積し，食生活の乱れや運動不足をユーザに通知する，といった健康管理のためのライフログアプリケーションが考えられる．

しかし，センサや機械学習による行動認識では，低次な行動情報しか認識できないという課題がある．例えば，加速度センサやマイク，GPS のデータを用いれば,「レストランで食事をした」というイベントは検出できるかもしれない．しかし,「何を食べたのか」や「誰と食事をしたのか」といった情報を認識することは難しい．先述のような健康管理アプリケーションに応用するには，ユーザがレストランの名前や料理名，一緒に食事をした人物の情報などを入力しなければならず利便性が低い．

より実用的なアプリケーションへ応用するためには，センサや機械学習による行動認識結果に加えて，その他の情報も組み合わせることで高次な行動情報を抽出できることが必要になる．例えば,「レストランで食事をした」という認識結果に加え，GPS から得られる位置情報を外部データと連携し，食事をしたレストランの名前や料理のカテゴリを機械的に統合することができれば利便性が高まる．

そこで本研究では，ライフログにおけるイベントをオントロジ [5] を用いて定義し，複合イベント処理 [6] を用いて高次な行動情報を抽出する基盤の構築を目的とする．また，ライフログにおける行動シナリオを想定し，高次なイベント検出の例を示すことにより，提案手法の妥当性を確認する．

本論文では，まず 2. で関連研究について述べた後，3. で高次な行動イベントを検出するためのオントロジについて説明する．さらに 4. では，3. で構築したオントロジに基づき，行動イベント処理を行うイベント処理システムについて述べる．5. では，ライフログを想定した高次な行動イベントの抽出例について説明する．6. では，構築したオントロジおよび行動イベント処理に関する考察を行う．最後に 7. で，本論文のまとめと今後の課題を述べる．

2. 関連研究

2.1 イベント処理に関連するオントロジ

イベントという概念をモデリングしたオントロジとして，Event Ontology [7] がある．Event Ontology において，Event という概念はほかの 5 つの概念と関係を持っている．それぞれ geo:SpatialThing（イベントが発生した場所），time:TemporalEntity（イベントが発生した時間），Thing（イベントを発生させた要因），foaf:Agent（イベントを発生させた主体），Thing（イベントによって生成された成果物）である．time:TemporalEntity および foaf:Agent は，それぞれ Time Ontology [8]，FOAF [9] で定義されている概念であり，EventOntology が新しく定義した概念ではない．詳細は後述する．Event Ontology では，Thing という概念が 2 回登場するが，factor および product という 2 つの関係が定義されており，別々の役割を果たす．したがって Event Ontology では，Event は「ある主体とある要因によって，ある時刻に，ある場所で発生し，それにより何かを生成するもの」として設計されている．本研究では，低次な行動イベントの定義を行う際に，この Event Ontology の設計を参考にしている．

Time Ontology は，時間に関する情報を表現するためのオントロジとして，World Wide Web Consortium（W3C）が提案している．Time Ontology では，時間を表現する概念として Instant や Interval などの概念が定義されている．Instant はある時刻を表現する概念であり，XML スキーマ (注1) で定義されている DateTime 型の時間情報（「2017-03-06T09:00:00」など）を保持している．Interval は期間を表す概念であり，開始時刻および終了時刻として Instant を保持している．Time Ontology は OWL（Web Ontology Language）[10] 形式のデータとして公開されているため，本研究ではこれを利用する．

また，FOAF（Friend of a Friend）は人物に関する情報を表現するための枠組みとして，Brickley らによって提案されている．FOAF では，人物を表現する概念として Person が定義されている．さらに，firstName や familyName，gender など人物についてより詳細に表現するための述語が用意されている．また，knows などの述語も定義されており，知人・友人関係を表現することも可能である．本研究の観点からすると，行動イベントを分析する際やデータを管理する際に，ある行動を誰が行ったのかという人物に関する情報を記述できると利便性が高まる．FOAF は RDF/XML 形式 (注2) として公開されており，本研究でもこれを利用する．

2.2 人間の行動を表現したオントロジ

人間の行動を表現したオントロジは数多く提案されており，モバイル端末のセンサを用いて行動分析を行うためのオントロジとして Riboni らの研究 [11] で構築されている ActivO オントロジがある．ActivO オントロジでは，主要な概念として Activity，SymbolicLocation，TimeExtent，Person，Artifact，CommunicationRoute が定義されている．すなわち ActivO オントロジは，人間の行動を，行動が行われた場所や時間，人物情報，行動と関連する人工物，通信方法を考慮して分析するように設計されている．

しかし，ActivO オントロジは，行動とセンサから得られるデータを直接結び付けておらず，高次な行動イベントをセンサなどから得られる低次な行動イベントの組合せで表現する本研究のアプローチには適さない．また，Riboni らはオントロジを統計的手法と組み合わせて行動認識することを想定しており，その点も本研究のオントロジおよびイベント処理を用いるアプローチとは異なっている．したがって本研究では，ActivO オントロジの主要な概念を参考にしつつ，新たに行動オントロジを構築した．オントロジの構築には，OWL を用いている．

3. 行動オントロジの概要

ここでは，新たに構築した行動オントロジについて説明する．まず，オントロジの主要な概念と概念間の関係について述べ，その後各概念について詳しく紹介する．

3.1 主要な概念と概念間の関係

本研究で構築したオントロジの主要な概念と，概念間の関係を図 1 に示す．

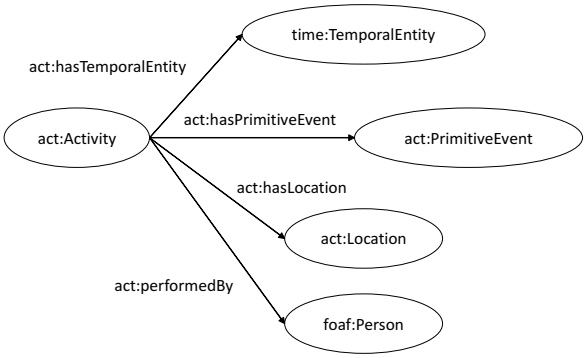

図 1: 主要概念の関係

本研究で構築したオントロジでは，主要な概念として高次な行動である act:Activity の他に act:PrimitiveEvent および act:Location を定義している．act:PrimitiveEvent は，モバイル端末などのセンサから得られる低次な行動イベントを，act:Location は位置情報を表す概念である．本研究で新しく定義した概念や関係は，接頭辞として「act:」を付与している．

一方で 2.1 節で述べたが，本研究では外部で定義されたオントロジも用いており，図 1 中の time:TemporalEntity および foaf:Person は，それぞれ Time Ontology および FOAF で定義されている概念である．接頭辞である「time:」は，TemporalEntity が Time Ontology で定義されている語彙であることを意味している．「foaf:」についても同様に，Person が FOAF で定義されている語彙であることを意味する．

act:Activity は time:TemporalEntity，act:PrimitiveEvent，および act:Location に対してそれぞれ act:hasTemporalEntity，act:hasPrimitiveEvent，および act:hasLocation という関係を持っている．すなわち，高次な行動イベントを，時間情報，低次な行動イベント，位置情報を考慮して抽出する．同様に foaf:Person に対しても act:performedBy という関係を持っており，行動を行った人間に関する情報も保持できるようになっている．

3.2 act:PrimitiveEvent

以下からは，主要な概念について詳細に説明する．

3.2.1 act:PrimitiveEvent とその他の概念の関連

act:PrimitiveEvent は，モバイル端末などのセンサから得られる低次な行動イベントを取り扱うために用いられる．以下の図 2 に act:PrimitiveEvent の定義を示す．

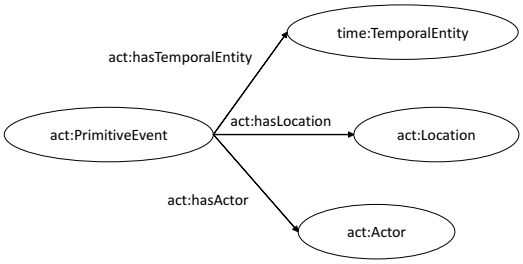

図 2: act:PrimitiveEvent

act:PrimitiveEvent は，time:TemporalEntity，act:Location，

(注1)：https://www.w3.org/TR/xmlschema-2/

(注2)：https://www.w3.org/TR/rdf-syntax-grammar/

foaf:Agent と，それぞれ act:hasTemporalEntity，act:hasLocation，act:hasActor という関係を持っている．すなわち，act:PrimitiveEvent は,「ある時刻に，ある場所で，何らかの主体よって生起するもの」として定義する．

3.2.2 PrimitiveEvent の下位概念

act:PrimitiveEvent は，下位概念として act:SoundEvent，act:LocationEvent，act:MotionEvent を定義している．

act:SoundEvent は音に関するイベントであり，act:Music（音が流れている），act:Talking（話している）などのイベントを定義している．

act:LocationEvent は位置情報に関するイベントであり，主にユーザの位置情報に関するイベントを表現する．act:LocationEvent では,「ある場所にいる」,「ある場所に滞在している」,「移動している」という 3 つの状況を想定し，Locate と Stay，Move を導入している.「ある場所にいる」,「ある場所に滞在している」というイベントは，act:LocateInHome（家にいる）や act:StayInRestaurant（レストランに滞在している）など具体的な場所と結びつけて表現し，移動は act:Moving という概念で表現される．

act:MotionEvent は人間の動作に関するイベントであり，act:Walking（歩いている），act:Sitting（座っている）などのイベントを定義している．

3.3 act:Activity

前述の通り，本研究で構築したオントロジでは高次な行動イベントを意味する概念として act:Activity を定義している．ここでは，act:Activity の具体例として「映画を見る」という行動を意味する act:WatchingMovie の定義を説明する．act:WatchingMovie は，以下のように定義される．

$$\texttt{WatchingMovie} \sqsubseteq \texttt{hasSoundEvent.Music} \sqcap \texttt{hasMotionEvent.Sitting} \sqcap \texttt{hasLocationEvent.StayInTheater}$$

act:WatchingMovie は，act:MotionEvent の act:Sitting（座っている），act:SoundEvent の act:Music（音が流れている），および act:LocationEvent の act:StayInTheater（映画館に滞在している）の組合せとして定義される．

3.4 act:Location

act:Location は，位置情報を表現するために用いられる．下位概念として act:Restaurant や act:Shop，act:Home などの象徴的な場所に関する情報に加え，act:Indoor や act:Outdoor などの屋内外に関する情報を取り扱うための概念を定義している．act:PrimitiveEvent の下位概念である act:LocationEvent は，この act:Location で定義された場所に関する概念を参照している．

4. オントロジに基づく行動イベント処理システム

ここでは，提案したオントロジに基づくイベント処理システムについて述べる．イベント処理システムには，以下の機能が求められる．

- 低次および高次な行動イベントを蓄積する機能
- 入力された低次なイベントに対してオントロジおよび推論規則を用いて高次な行動イベントを生成する機能
- 蓄積されたイベントに対して問合せする機能

本論文では，上記の機能を単体で簡潔に実装できることから，GraphDB [12] を用いたシステムの構築例を紹介する．GraphDB は，Ontotext 社 (注3) が開発・提供している RDF データストアであり，下記のような特徴を備えている．

- RDF データの格納
- Web ブラウザベースのデータベース管理インタフェースの提供
- SPARQL を用いた問合せへの対応
- オントロジおよび独自の推論ルールを利用した推論機能
- アプリケーション開発用のライブラリの提供

GraphDB では，Free，Standard，Enterprise，Database-as-a-Service，Cloud on AWS の 5 つのエディションが提供されており，本研究では Free エディションを使用する．

4.1 システムの全体像

ここでは，行動イベント処理システムの全体像について説明する．以下の図 3 に，システムの構成要素とそれらの関連を示す．

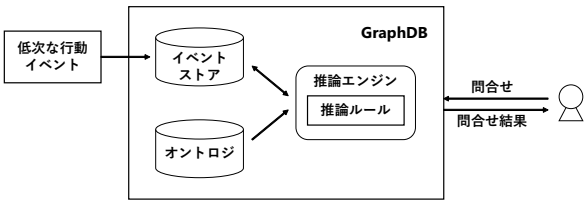

図 3: 行動イベント処理システムの全体像

行動イベント処理システムは，入力として低次な行動イベントを受け取り，イベントストアに格納する．続いて，推論エンジンおよびオントロジにより，高次な行動イベントが推論，生成され，イベントストアに格納される．推論エンジンには，独自に定義した推論ルールが適用されている．ユーザはイベント処理システム内に蓄積された低次および高次な行動イベントに対し問合せを行うことにより，イベント処理結果を受け取ることができる．これらの低次な行動イベントの蓄積，高次な行動イベントの生成，蓄積された行動イベントに対する問合せは，GraphDB フレームワークを用いて実装している．

以下からは，それぞれの構成要素について説明する．

4.1.1 イベントストア

イベントストアでは，低次な行動イベントおよび高次な行動イベントを蓄積する．低次な行動イベントは，センサなどにより断続的に生成され，イベントストアに格納される．ここで，低次な行動イベントへの整形はシステム入力前に行われていることを想定し，本論文では議論しない．

また，高次な行動イベントは，入力された低次な行動イベン

(注3)：http://www.ontotext.com

トに対して推論エンジンによって生成され，イベントストアに格納される．

4.1.2 オントロジ

オントロジは，推論エンジンでの処理に用いられる．3. で述べたオントロジで定義された概念や関係などの語彙は，推論エンジンで高次な行動イベントを生成する際に用いられる．

また，GraphDB ではオントロジは RDF 形式のデータとして保持されており，RDF/XML 形式や Turtle 形式 (注4) など様々な形式で表現された RDF データを扱うことができる．そのため，LOD（Linked Open Data）[13] として RDF 形式で公開されているオントロジやデータセットを統合し，推論を拡張することが容易である．

4.1.3 推論エンジン

推論エンジンでは，オントロジおよび独自に定義した推論ルールを用いて高次な行動イベントを生成する．これまで述べたとおり，オントロジには対象とする世界の概念および概念の実体，概念間の関係を定義できる．しかし,「低次イベント A と低次イベント B が同時に生起した場合，新しく高次イベント C を生成する」といった複数の低次なイベントを組み合わせて推論を行うための規則は記述できない．

そこで，オントロジ以外の規則に基づき任意の推論を行うための枠組みを導入する必要がある．本研究では，GraphDB に実装されている，独自の推論ルールを記述し推論エンジンに適用する機能を用いる．

図 4 に推論ルールの記法について示す．推論ルールは，推論ルールの Id，前提部，帰結部から構成され，前提部に記述された条件が満たされた場合に帰結部に記述されたデータが生成される．Id は推論ルール冒頭に記述され，前提部と帰結部の区別は------を用いて行う．すなわち図中では，Id は 1 行目，前提部は 2 行目および 3 行目，帰結部は 5 行目に相当する．また，<>に囲まれていない小文字のアルファベットは変数を意味する．つまり，a，b，および x は変数を表す．

図 4 では，まず 2 行目で変数 a が変数 b のサブクラス（下位概念）であることを記述している．すなわち，a および b は概念である．さらに 3 行目で，x は型として a を持っていることを記述している．したがって，x は a の個体である．そしてこの条件のもと，5 行目で x は型として b を持つ，というデータを生成するように記述している．まとめるとこの推論ルールでは,「a が b の下位概念かならば，x は a の個体である」という推論を実行するルールとなる．

```
1   Id: subClassOf
2     a <rdfs:subClassOf> b
3     x <rdf:type> a
4   ------------------------------------
5     x <rdf:type> b
```

図 4: 推論ルールの例

(注4) : https://www.w3.org/TR/turtle/

4.2 イベント処理の例

ここでは，イベント処理システムにおける行動イベント処理の例について述べる．高次な行動イベントの生成は，入力された低次な行動イベントに対して，推論ルールを適用した推論エンジンによって行われる．

4.2.1 検出するイベントの例

行動イベント処理について，前述の act:WatchingMovie の検出を例に説明する．まず，イベント処理システムに以下の表 1 に示す低次な行動イベントが入力されたものとする．

表 1: 入力する低次なイベントデータ

時間	イベント名	上位概念
2016-11-21T09:00:00	act:Music	act:SoundEvent
2016-11-21T09:00:00	act:Sitting	act:MotionEvent
2016-11-21T09:00:00	act:StayInTheater	act:LocationEvent

3. 3 節で説明したオントロジの定義から，表 1 の入力データは，時刻 2016-11-21T09:00:00 において act:WatchingMovie の定義を満たす．したがって，推論によって「時刻 2016-11-21T09:00:00 に行動 act:WatchingMovie が行われた」というイベントが生成される．

4.2.2 低次な行動イベントの実体

ここでは，入力される低次な行動イベントの実体について説明する．act:Music，act:Sitting，act:StayInTheater の上位概念はそれぞれ act:SoundEvent，act:MotionEvent，act:LocationEvent であり，さらにそれらの上位概念は act:PrimitiveEvent である．act:PrimitiveEvent は，time:TemporalEntity と act:hasTemporalEntity という関係を持っており，その関係は下位概念にも継承される．したがって，act:Music，act:Sitting，act:StayInTheater も time:TemporalEntity との間に act:hasTemporalEntity という関係を持つ．

まず time:TemporalEntity の実体について説明する．図 5 に，time:TemporalEntity を RDF 形式で表現したものを示す．1 行目は act:TimeInstance1 が個体であることを定義しており，OWL の語彙を用いている．2 行目では act:TimeInstance1 の型が time:Instant であることを定義している．time:Instant は，time:TemporalEntity の下位概念であり，そして 3 行目で，act:TimeInstance1 が具体的な時間の値として「2016-11-21T09:00:00」をもっていることを表現している．

```
1    act:TimeInstance1  rdf:type  owl:NamedIndivisual;
2         rdf:type  time:Instant ;
3         time:inXSDDateTime  2016-11-21T09:00:00.
```

図 5: time:Instant の実体

続いて，act:Music の実体の定義を行う．act:Music の実体の例を図 6 に示す．time:Instant の実体と同様に 1，2 行目で act:Music1 が act:Music の実体であることを定義している．さらに 3 行目では，act:Music1 が時間要素とし

て act:TimeInstance1 を持っていることを表現している．act:Sitting および act:StayInTheater の実体の定義も，図 6 と同様に行う．

```
act:Music1  rdf:type  owl:NamedIndivisual;
      rdf:type  act:Music ;
      act:hasTemporalEntity act:TimeInstance1.
```

図 6: act:Music の実体

4.2.3 推論ルール

図 7 に示すのは，複数の低次な行動イベントを組み合わせて act:WatchingMovie を検出するための推論ルールである．図中の例では，2 行目は a が act:Music の個体であることを意味している．同様に，4，6 行目では b および c が，それぞれ act:Sitting，act:StayInTheater の個体であることを示している．3 行目では，a が time:Instant の個体を保持していることを指し，5，7 行目では b および c も a と同じ time:Instant の実体 t を保持しているという制約を表している．すなわち，上記の推論ルールでは，act:Music，act:Sitting，および act:StayInTheater が同時に観測された場合，新しく 9 行目を追加するという推論ルールになっている．

```
1  Id: WatchingMovie
2    a <rdf:type> <act:Music>
3    a <act:hasTemporalEntity> t
4    b <rdf:type> <act:Sitting>
5    b <act:hasTemporalEntity> t
6    c <rdf:type> <act:StayInTheater>
7    c <act:hasTemporalEntity> t
8  ------------------------------------
9    t <act:hasActivity> <act:WatchingMovie>
```

図 7: act:WatchingMovie の推論ルール

4.2.4 問合せの例

ここでは，推論ルールによって行動イベント処理が行われていることを，SPARQL [14, 15] 問合せを用いて確認する．SPARQL は，RDF 形式のデータに対する問合せ言語であり，関係データベース管理システムで用いられる問合せ言語 SQL に似た文法を持っている．SELECT 句で問い合わせたい RDF トリプルのリソースを指定し，WHERE 句にその条件を記述することができる．

図 8 に，act:WatchingMovie を行った時間を問い合わせる SPARQL 問合せを示す．5 行目では，図 7 で示した推論ルールに基づき，act:WatchingMovie が推論された際の time:Instant の実体を条件として与えている．さらに 6 行目では，5 行目で与えた time:Instant の実体が保持する時間の値を問い合わせている．

図 9 は，図 8 の SPARQL 問合せを行動イベント処理システムで実行した結果である．「— Load Data—」以降が，図 8 の SPARQL 問合せにより出力されている部分である．まず，instance として「http://onto.nagoya-u.ac.jp/Activity#TimeInstance1」が出力されており，図 5 で定義

```
1  PREFIX act: <http://onto.nagoya-u.ac.jp/Activity#>
2  PREFIX time: <http://www.w3.org/2006/time#>
3  SELECT ?instance ?time
4  WHERE{
5    ?instance act:hasActivity act:WatchingMovie.
6    ?instance time:inXSDDateTime ?time.
7  }
```

図 8: act:WatchingMovie を問い合わせる SPARQL 問合せ

された time:TemporalEntity の実体と一致する．また，time として，「2016-11-21T09:00:00」が出力されており，これも図 5 で示した時間の値と一致する．値の後ろに付与された「^^<http://www.w3.org/2001/XMLSchema#String>」は，値「2016-11-21T09:00:00」が XML スキーマで定義された String 型であることを示している．

これにより，表 1 に示した入力データに対して，図 7 で示した推論ルールが適用され，SPARQL 問合せにより act:WatchingMovie を行った時間の検出ができることを示せた．

図 9: act:WatchingMovie 問合せの実行結果

5. 高次な行動イベントの抽出例

ここでは，これまでに示した行動イベント処理手法を用いて高次な行動イベントを抽出する例を示す．具体的には，3. で説明したオントロジで定義されている概念を用いてライフログアプリケーションで休日の行動を記録することを想定し，高次な行動イベントの抽出を行う．

5.1 行動イベント処理のシナリオ

表 2 に，行動を記録する一日のシナリオを示す．シナリオでは，映画を観る，徒歩で移動する，レストランで食事をする，車で移動する，店で買い物をする，というあるユーザの休日の行動を想定する．

表 2: 行動シナリオ

時間	行動
09:00–11:00	映画を観る
11:00 – 12:00	徒歩で移動する
12:00 – 14:00	レストランで食事をする
14:00 – 15:00	車で移動する
15:00 – 17:00	店で買い物をする

表 3 に，シナリオに従って行動した際に検出される低次な行動イベントを示す．なお，表 3 中では接頭辞の「act:」は省略している．表 2 の行動シナリオに従って，act:SoundEvent，act:MotionEvent，act:LocationEvent を 1 時間ごとに人工的に生成し，行動イベント処理に用いる．

表 3: 入力する低次なイベントデータ

時間	SoundEvent	MotionEvent	LocationEvent
2016-11-21T09:00:00	Music	Sitting	StayInTheater
2016-11-21T10:00:00	Music	Sitting	StayInTheater
2016-11-21T11:00:00	Music	Sitting	StayInTheater
2016-11-21T12:00:00	Loud	Walking	Moving
2016-11-21T13:00:00	Talking	Eating	StayInRestaurant
2016-11-21T14:00:00	Talking	Eating	StayInRestaurant
2016-11-21T15:00:00	InCar	Sitting	Moving
2016-11-21T16:00:00	Crowded	Walking	StayInShop
2016-11-21T17:00:00	Crowded	Walking	StayInShop

5.2 行動定義

ここでは，前述のシナリオの中で行った行動を検出するためのオントロジでの行動定義について述べる.「映画を観る」という行動は，3.3 節で説明した定義と同様のものを用いる.

まず「徒歩で移動する」という行動を，下記のように定義する.

$$\texttt{MovingByWalk} \sqsubseteq \texttt{hasMotionEvent.Walking} \sqcap \texttt{hasLocationEvent.Moving}$$

act:MovingByWalk を，act:MotionEvent の act:Walking（歩いている）と act:LocationEvent の act:Moving（移動している）の組合せとして定義する.

続いて,「レストランで食事をする」という行動を，下記のように定義する.

$$\texttt{EatInRestaurant} \sqsubseteq \texttt{hasSoundEvent.Talking} \sqcap \texttt{hasMotionEvent.Eating} \sqcap \texttt{hasLocationEvent.StayInRestaurant}$$

act:EatInRestaurant を，act:SoundEvent の act:Talking（話している），act:MotionEvent の act:Eating（食事をしている），そして act:LocationEvent の act:StayInRestaurant（レストランに滞在している）の組合せとして定義する.

さらに,「車で移動する」という行動の場合は下記のように定義する.

$$\texttt{MovingByCar} \sqsubseteq \texttt{hasSoundEvent.InCar} \sqcap \texttt{hasMotionEvent.Sitting} \sqcap \texttt{hasLocationEvent.Moving}$$

act:MovingByCar を，act:SoundEvent の act:InCar（車内にいる），act:MotionEvent の act:Sitting（座っている），そして act:LocationEvent の act:Moving の組合せとして定義する.

最後に,「店で買い物をする」という行動を下記のように定義する.

$$\texttt{Shopping} \sqsubseteq \texttt{hasSoundEvent.Crowded} \sqcap \texttt{hasMotionEvent.Walking} \sqcap \texttt{hasLocationEvent.StayInShop}$$

act:Shopping を，act:SoundEvent の act:Crowded（混雑している），act:MotionEvent の act:Walking，そして act:LocationEvent の act:StayInShop（店に滞在している）の組合せとして定義する.

上述のように，3. で説明したオントロジを用いることで，アプリケーションで検出する行動イベントを簡易に定義することができる．これらの定義を用いて，行動イベントの検出を行う.

5.3 推論ルールの構築

ここでは，推論ルールについて説明する．図 10 に，act:MovingByWalk の推論ルールを示す．まず図 10 の推論ルールの前提部について説明する．推論ルールの 2 行目では，a が act:Walking 型であること，つまり a が act:Walking の個体であることを示しており，4 行目も同様である．また，3 行目では a が t との間に act:hasTemporalEntity という関係があることを示している．オントロジの定義では，act:Walking は act:MotionEvent および act:PrimitiveEvent の下位概念であり，act:PrimitiveEvent は time:TemporalEntity との間に act:hasTemporalEntity という関係を持っている．したがって，t は time:TemporalEntity の個体を意味する．3 行目および 5 行目で，a および b が同じ個体を持っていることを記述しているため，図 10 の推論ルールの前提部では，act:Walking および act:Moving が同時に発生した場合を指定している.

続いて帰結部について説明する．まず 7 行目では，型が act:MovingByWalk である新しいリソース d を生成している．さらに 8 行目で，d に「act:WatchingMovie」というラベルを与えている．このラベルは，問合せの際に用いられる．そして 9 行目で，d に時間の要素として t を与えている．帰結部をまとめると，act:MovingByWalk 型のリソース d を生成し，「act:MovingByWalk」というラベルと，t という時間要素を与えるという推論ルールとなっている.

```
1    Id: MovingByWalk
2      a <rdf:type> <act:Walking>
3      a <act:hasTemporalEntity> t
4      b <rdf:type> <act:Moving>
5      b <act:hasTemporalEntity> t
6    -----------------------------------
7     d <rdf:type> <act:MovingByWalk>
8     d <rdfs:label> <act:MovingByWalk>
9     d <act:hasTemporalEntity> t
```

図 10: MovingByWalk の推論ルール

act:EatInRestaurant，act:MovingByCar，act:Shopping の推論ルールについても，図 10 同様に定義しているため説明を省略する．帰結部の定義は同じものを用いており，前提部の定義が 5.2 節で示した行動定義に従ったものになる.

5.4 イベント処理の実行結果

図 11 に，推論によって生成された高次な行動イベントを検出するための SPARQL 問合せを示す．ここでは，高次な行動イベントが発生した時間および行動のラベルを問い合わせている．WHERE 句内の 7，8 行目では，型が act:Activity（高次な行動イベント）であるようなリソースを指定し，そのリソースのラ

ベルが変数?label として問合せ結果に出力される．また，9 行目では 7 行目で指定したリソースが持つ time:TemporalEntity の個体を参照し，10 行目で時間情報を問合せ結果に出力している．11 行目の「ORDER BY ?time」は，検索結果を変数?time の値で整列させている．また，行動イベント処理システム上で図 11 の SPARQL 問合せを実行した結果を図 12 に示す．これにより，表 3 に示した入力イベントをもとに推論が行われ，高次な行動イベントが生成されていることがわかる．

```
1  PREFIX act: <http://www.example.org/onto>
2  PREFIX time: <http://www.w3.org/2006/time#>
3  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
4  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5  SELECT ?time ?label
6  WHERE{
7    ?activity rdf:type act:Activity.
8    ?activity rdfs:label ?label.
9    ?actiyity act:hasTemporalEntity ?t.
10   ?t time:inXSDDateTime ?time.
11 }  ORDER BY ?time
```

図 11: Activity を問い合わせる SPARQL 問合せ

```
<終了> RulesetTest [Java アプリケーション] C:¥pleiades¥java¥8¥bin¥javaw.exe (2017/01/26 16:01:45)
Current file: ../../configs/rules/act_test.pie
Compiled: 'C:\Users\Ryo Nakamura\Documents\GraphDB\WIn_compile_try\graphdb-
--- Load Data ---
time = "2016-11-21T09:00:00"^^<http://www.w3.org/2001/XMLSchema#string>
label = http://onto.nagoya-u.ac.jp/Activity#WatchingMovie
time = "2016-11-21T10:00:00"^^<http://www.w3.org/2001/XMLSchema#string>
label = http://onto.nagoya-u.ac.jp/Activity#WatchingMovie
time = "2016-11-21T11:00:00"^^<http://www.w3.org/2001/XMLSchema#string>
label = http://onto.nagoya-u.ac.jp/Activity#WatchingMovie
time = "2016-11-21T12:00:00"^^<http://www.w3.org/2001/XMLSchema#string>
label = http://onto.nagoya-u.ac.jp/Activity#MovingByWalk
time = "2016-11-21T13:00:00"^^<http://www.w3.org/2001/XMLSchema#string>
label = http://onto.nagoya-u.ac.jp/Activity#EatInRestaurant
time = "2016-11-21T14:00:00"^^<http://www.w3.org/2001/XMLSchema#string>
label = http://onto.nagoya-u.ac.jp/Activity#EatInRestaurant
time = "2016-11-21T15:00:00"^^<http://www.w3.org/2001/XMLSchema#string>
label = http://onto.nagoya-u.ac.jp/Activity#MovingByCar
time = "2016-11-21T16:00:00"^^<http://www.w3.org/2001/XMLSchema#string>
label = http://onto.nagoya-u.ac.jp/Activity#Shopping
time = "2016-11-21T17:00:00"^^<http://www.w3.org/2001/XMLSchema#string>
label = http://onto.nagoya-u.ac.jp/Activity#Shopping
```

図 12: Activity 問合せの実行結果

6. 考　　察

5. での結果から，3. で構築したオントロジおよび 4. で説明した行動イベント処理システムの妥当性を確認することができた．ここでは，これまでに述べた行動イベント分析手法についての考察を行う．

6.1 提案手法の利点

本研究では，センサや機械学習の分類モデルなどから得られる行動認識情報を低次な行動イベントとして表現し，低次な行動イベントを組み合わせて高次な行動イベントを定義している．本研究で構築した手法は，機械学習で行動認識を行う方法に比べ，イベントの再利用性および拡張性の観点で優れていると考えられる．

例えば，機械学習を用いて「映画を観ている」という行動を検出しようとする場合には，分類モデルの作成に大きなコストがかかり，モデルの再利用性も低い．なぜなら，単体のセンサから得られる情報だけで「映画を観ている」という複雑な行動を判別することは難しく，複数のセンサデータを用いてモデルを学習させる必要があるからである．さらに学習させたモデルは，「映画を観ている」という行動を検出するためにしか用いることができない．

一方で，本研究は複雑な行動でも act:PrimitiveEvent の組合せとして表現でき，低次な行動イベントや高次な行動イベントの再利用性が高い．3. で説明したオントロジでは，「映画を観ている」という行動を「act:Sitting」「act:Music」「act:StayInTheater」の組合せで表現しているが，「act:Sitting」は，「act:MovingByCar」という行動イベントの定義にも用いられている．さらに，検出された高次な行動イベントを用いて，新たに別の行動イベントを定義することも可能である．

また，任意の行動イベント処理への対応が容易であるという，拡張性の高さも本研究の利点である．提案手法では，オントロジと推論ルールを拡張することで，任意の行動イベント処理に対応できる．イベント処理に用いているのはオントロジと推論ルールのみなので，一から分類モデルを生成する必要がある機械学習のアプローチに比べ，専門知識を持たないユーザでも自由に拡張できる可能性がある．

6.2 オントロジの拡張

一方で，3. で説明したオントロジに関して，検討が必要な部分も考えられる．具体的には，時間の表現および位置情報の表現である．

これまで示した推論ルールでは，低次な行動イベントの発生時刻が一致していることを条件として，高次な行動イベントを生成している．しかしこのような推論ルールでは，データが増加した際に処理するイベントの数も増加し，イベント処理システムの性能に影響が出る可能性がある．したがって，時間を時刻（タイムスタンプ）ではなく期間（インターバル）で表現したほうが効率的な場合が考えられる．そのため，低次な行動イベントの時間情報をインターバルとして受け取り，オーバーラップしている期間だけ高次な行動イベントを生成するなど，時間の表現がインターバルに対応できるようにオントロジを拡張する必要がある．

また，本研究では，位置に関する情報を act:Location で表し，低次な行動イベントの act:LocationEvent が act:Location を参照する形式で表現している．このように表現することで，高次な行動イベントを簡易かつ直観的に表現することを意図している．しかし，より汎用性の高い位置情報の表現方法を用いることで，より効率的な処理を実現できると考えられる．例えば，GraphDB では OGC（Open Geospatial Consortium）(注5) の GeoSPARQL [16] プラグインが同梱されており，GeoSPARQL が用いている Simple Features モデルなどが利用できる．Simple Feartures モデルでは，地理関係や空間情報を SpatialObject という概念で表現し，さらに緯度，経度や空間の形などを Geometry という下位概念で表現できる．そして Simple Feartures モデルで表現したデータを，GeoSPARQL を使って問い合わせることができる．このような汎用性の高い位置情報に関する枠組みを導入しつつ，同時に高次なイベントを

(注5)：http://www.opengeospatial.org/

簡易に記述できるような表現方法を検討する必要がある．

6.3 行動イベント処理システムの実装方法

ここでは，イベント処理システムの構成要素として GraphDB を用いることについて考察する．

GraphDB は，RDF データの管理や推論，SPARQL による問合せ機能を単体で実装することが可能である．一方で，入力されるデータの種類によっては，GraphDB による実装が最適でない場合が考えられる．GraphDB は，新しいデータが入力されるたびに推論が実行されるという仕様になっている．すなわち，データが高頻度で入力された場合，頻繁に推論が行われることでシステムの処理性能が低下する，といったことが考えられる．

したがって，まずは入力イベント数と処理性能の相関などの評価を行い，それに基づいてイベント処理システムの構成について検討する必要がある．上述のような要求には，C-SPARQL [17] のような RDF ストリームに対応した枠組みを用いることで対応できると考えられる．断続的なデータへの対応は C-SPARQL で行い，定期的に GraphDB と同期することでストリームを効率的に処理できる可能性がある．

また，イベント処理に推論ルールを用いることに関しても検討が必要である．本研究では，推論ルールを用いることで任意の行動イベント処理を実現している．GraphDB で用いられる推論ルールの文法も直観的なため，ユーザが容易に拡張できるという利点がある．

一方で，複雑なイベントに対する推論ルールの表現力は，十分に検討する必要があると考えられる．例えば，「特定のイベントが順番に生起する場合」などを推論ルールで表現することは難しい．なぜなら，推論ルールの前提部に記載された条件が論理積で処理されるため，条件の発生順序を保持できないからである．さらに，実際の数値を用いて比較するような場合なども表現が困難である．例えば，「ある地域の半径 1km 以内に入った場合」などである．GraphDB の推論ルールでは，このような数値に基づく比較を表現することができない．

したがって，推論ルールによるイベント処理の表現力を十分に吟味する必要がある．ただし，オントロジおよび問合せ手法を拡張することでこのような問題を解決できる可能性もある．GeoSPARQL では，地理的な距離に基づいて情報を選別するなどの処理を容易に記述できる．このような，オントロジの拡張や外部の枠組みの利用も考慮した上で，推論ルールに基づくイベント処理手法を検討する必要がある．

7. おわりに

本研究では，低次な行動イベントを組み合わせ，高次な行動イベントを抽出するためのオントロジを構築した．既存の機械学習などに基づく行動認識手法では，認識できる行動情報が低次であるという課題がある．そこで，オントロジを用いて高次な行動情報を，低次な行動イベントの組合せとして定義し，イベント処理を用いて高次な行動イベントを抽出する手法を提案した．本論文では，行動オントロジの構築に加え，行動イベント処理システムを GraphDB を用いて実装した．さらに，実装した行動イベント処理システムに基づき，ライフログアプリケーションを想定した高次な行動イベントの抽出を行うことにより，提案手法の妥当性を考察した．

今後の課題として，オントロジの拡張や推論ルールの追加があげられる．また，構築した行動イベント処理システムの機能充実も検討する必要がある．

謝　辞

本研究は，JST COI（Center of Innovation）プログラムおよび科研費（16H01722, 26540043）による．

文　献

[1] 上坂大輔, 村松茂樹, 岩本健嗣, 横山浩之. 手に保持されたセンサを用いた歩行者向けデッドレコニング手法の提案. 情報処理学会論文誌, Vol. 52, No. 2, pp. 558–570, 2011.

[2] 大内一成, 土井美和子. スマートフォンを用いた生活行動認識技術. 東芝レビュー, Vol. 68, No. 6, pp. 40–43, 2013.

[3] Oscar D. Lara and Miguel A. Labrador. A Survey on Human Activity Recognition using Wearable Sensors. *IEEE Communications Surveys and Tutorials*, Vol. 15, No. 3, pp. 1192–1209, 2013.

[4] Andres Bulling, Blanke Ulf, and Bernt Schiele. A Tutorial on Human Activity Recognition Using Body-Worn Inertial Sensors. *ACM Computing Surveys (CSUR)*, Vol. 46, No. 3, pp. 33:1–33:33, 2014.

[5] 溝口理一郎. 知の科学 オントロジー工学の理論と実践. オーム社, 2012.

[6] Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR)*, Vol. 44, No. 3, pp. 15:1–15:62, 2012.

[7] Yves Raimond and Samer Abdallah. The event ontology. 2007. `http://motools.sourceforge.net/event/event.html`.

[8] Jerry R Hobbs and Feng Pan. Time Ontology in OWL. *World Wide Web Consortium*, 2016. `http://www.w3.org/TR/owl-time/`.

[9] Dan Brickley and Libby Miller. FOAF vocabulary specification 0.99. 2014. `http://xmlns.com/foaf/spec`.

[10] Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language. *World Wide Web Consortium*, 2004. `http://www.w3.org/TR/owl-features/`.

[11] Daniele Riboni and Claudio Bettini. COSAR: hybrid reasoning for context-aware activity recognition. *Personal Ubiquitous Computing*, Vol. 15, No. 3, pp. 271–289, 2011.

[12] GraphDB. `http://graphdb.ontotext.com`.

[13] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, Vol. 5, No. 3, pp. 1–22, 2009.

[14] Steve Harris and Andy Seaborne. SPARQL 1.1 Query Language. *World Wide Web Consortium*, 2013. `http://www.w3.org/TR/sparql/11-query`.

[15] 加藤文彦, 川島秀一, 岡別府陽子, 山本泰智, 片山俊明. オープンデータ時代の標準 WebAPI SPARQL. インプレス R&D, 2015.

[16] Robert Battle and Dave Kolas. Enabling the geospatial semantic Web with Parliament and GeoSPARQL. *Semantic Web*, Vol. 3, No. 4, pp. 355–370, 2012.

[17] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. C-sparql: Sparql for continuous querying. *Proceedings of the 18th International Conference on World Wide Web*, pp. 1061–1062, 2009.