

AN ONTOLOGY-BASED APPROACH FOR ACTIVITY RECOGNITION FROM SENSORS IN OFFICES

By Xueying Gu

Project Advisor: Dr. Silvia Nittel

Co-Advisor: Dr. Torsten Hahmann

An Abstract of the Project Presented

in Partial Fulfillment of the Requirements for the

Degree of Master of Science

December, 2015

Understanding the behavior of people in a building can be useful to save energy in buildings and improve efficiency of employees. To study the behavior of people in office buildings, sensors can be used to collect raw data about people's presence in rooms and their movement through buildings. However, to understand higher-level behavioral patterns, we first developed an application ontology, written in Common Logic, that describes this information of interest at a higher level of abstraction. For example, which characteristics constitute a meeting? How are they related to movement data? Following, we translate the defined ontological concepts into MySQL scripts and Java algorithms to 'mine' the low-level sensor data stored in a relational database system. With the programs, we can retrieve some important information about offices such as: how frequently are offices and shares spaces utilized? How many meetings take place on average during a week? A number of use cases demonstrate the capabilities of the system.

ACKNOWLEDGEMENT

I would like to extend my sincere gratitude to all my friends, family and the School of Computing and Information Science . Especially I would like thank my advisor Dr. Silvia Nittel for her guidance and support throughout the past two years. I also want to thank my Co-Advisor, Dr. Torsten Hahmann for his strict attitude. They pushed me to finish this project. Without their valuable inputs, this project would not have not obtained its fruitful result. I shall also thank Mark Plummer and JC Whittier. I like to thank Professor Kate Beard and Harlan Onsurd for serving as my committee members and other professors who have impacted me in the past years.

I am grateful to my roommates Lu Wang, Yan Liu and Jing Chen who are very kind and friendly. I also appreciate the care and help from my house lord – Richard Hill. I want to thank my parents and my girlfriend. Last but not the least, I thank the Graduate School at the University of Maine.

Table of Contents

Chapter 1: INTRODUCTION.....	1
1.1 Motivation.....	1
1.2 Project Objectives	2
1.3 Organization	2
Chapter 2: BACKGROUD AND RELATED WORK	3
2.1 Ontology	3
2.2 Sensor Stream	3
2.3 Related Work	4
Chapter 3: ONTOLOGY FOR HUMAN ACTIVITY	6
3.1 Original Data	6
3.2 Domain Analysis	7
3.3 Multiple Person Colocation Event	8
3.4 Vocabulary.....	9
3.5 Relationship.....	11
3.6 Conceptual Model	12
3.7 Axioms.....	15
3.7.1 Subsumption Relations	16
3.7.2 Time.....	17
3.7.3 Multiple Person Colocation Event	19
3.7.4 Meeting	21
3.8 Ontology Evaluation.....	22
3.8.2 Consistency	23
3.8.1 Competency.....	24
Chapter 4: DATA ANALYSIS AND ALGORITHMS.....	28
4.1 Single Person Colocation Event.....	28
4.2 2-person Colocation Events	29
4.3 Multiple Person Colocation Events.....	31
4.4 Pseudo Code of Generating Multiple Person Colocation Event	33
4.5 Meeting Classification	34
4.6 Verification	35

Chapter 5: CONCLUSIONS AND FUTURE WORK.....	38
BIBLIOGRAPHY	39
Appendix	41
Script of generating 2-person colocation event.....	41
Script of generating multiple person colocation event	43
Code of Node	47
Test Script.....	48
Module of Activity	52
Module of Person.....	64
Module of AllRelation	65
Module of TemporalEntity	70
Module of Equipment	77
Consistency Result	78

LIST OF FIGURES

Figure 3.1.....	6
Figure 3.2.....	7
Figure 3.3.....	8
Figure 3.4.....	12
Figure 3.5.....	13
Figure 3.6.....	13
Figure 3.7.....	14
Figure 3.8.....	14
Figure 3.9.....	15
Figure 3.10	22
Figure 4.1.....	27
Figure 4.2.....	28
Figure 4.3.....	29
Figure 4.4.....	29
Figure 4.5.....	30
Figure 4.6.....	31
Figure 4.7.....	31
Figure 4.8.....	31
Figure 4.9.....	36

Chapter 1: INTRODUCTION

1.1 Motivation

Today, there is significant research and commercial interest in creating ‘smart buildings’. Smart buildings help saving energy [1] and assist occupants through different services. For energy saving, questions of interest are – “when are lights on and off?”, “When, where and how was temperature in the building adjusted?” For assistance purposes, questions are “Which rooms are available for meetings right now?”, or “Was there a team meeting taking place in Office 101 at noon yesterday?”. To understand and control smart buildings, understanding human activity is an important factor. To answer the questions above, we need to be aware of human movement and activity to adjust or predict e.g. energy usage.

To study human activity, people place sensors in buildings to collect raw data about human activity. We are able to collect massive amounts of sensor data streams today with novel technology. But how can we understand what they capture? Can we know what activity is happening in a certain room? How can we understand sensor data in human and computer interpretable form?

There are two types of techniques that help us understand raw sensor data - data-driven and knowledge-based techniques. Data-driven methods, also known as quantitative methods, mainly use statistics and machine learning to understand and fuse sensor data. These methods are good at dealing with noise and uncertain data [2]. However, machine learning classifiers may represent objects in terms of a low-level features, which have little inherent meaning to a human examiner and are difficult to aggregate into higher-level abstractions. We might also not have collected sufficient amounts of data that machine learning needs. Knowledge-based methods, on the other hand, build concise representations of concepts that are appropriate for human understanding as well as logic based reasoning and develop algorithms to aggregate raw data towards those knowledge representations. Usually a knowledge-based method has two sub-systems: a knowledge

base, often in the form of an ontology, and a reasoning engine. Knowledge-based methods have already been applied in pervasive environment and improve interoperability [3]. For this project, we chose a knowledge-based approach to model human activity in buildings, and develop algorithms to aggregate raw data into those concepts.

1.2 Project Objectives

The project develops an application ontology [4] to describe basic human movement and 'higher-level' aggregated knowledge such as *meetings* and *types of meetings*. There are mainly two objectives in this project. The first objective is building ontology for human behaviors in buildings. The second objective is translating ontology axioms into algorithms with MySQL scripts and Java code to understand raw sensor data and extract those higher-level knowledge concepts from the data. Finally, based on data from Autodesk, we test whether our algorithms are correct and consistent with the ontology. At the same time, we need to consider the time and memory efficiency of algorithms.

1.3 Organization

In this paper, Chapter 2 is about related work and background. Chapter 3 is about the construction of the application ontology. Chapter 4 describes the algorithms to create multiple person colocation events and extract meeting information questions. Chapter 5 discusses the conclusions and future direction of the work.

Chapter 2: **BACKGROUD AND RELATED WORK**

2.1 Ontology

An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy [5]. In computer science, an ontology consists of a standardized vocabulary and definitions of classes, attributes and interrelationships of the classes and entities that really or fundamentally exists in a specific domain [6].

This project studies the particular domain of human activities in offices. We use an approach based on an application ontology, which is a lightweight ontology tailored to a specific task or application. It supplies minimal terminology for the needs of a specific community. It is lightweight, meaning that it may not take the form of fully-fledged axiomatic theory. It may just be a taxonomy about specific domain. Besides that, it does not necessarily require representational accuracy.

The application ontology is written in the CLIF syntax of the Common Logic standard [7]. Common Logic consists of a family of first order logic languages. It is intended to facilitate the exchange and transmission of knowledge in computer science. The reason why we chose Common Logic rather than OWL [8] is that it is more expressive. For example, OWL cannot directly express relationship that involve three entities but Common Logic can. Furthermore, OWL is limited in its ability to formalize hierarchies of relationships. Another reason is that both the SQL language and Common Logic are first-order languages. So Common Logic can express SQL very well.

2.2 Sensor Stream

A sensor stream is a time series of sensor measurement $ms_j = \langle id, t_i, ls_j, v_1, v_2, \dots, v_n \rangle$ generated by a sensor node with id , based on one or more attached sensors. To get sensed values, we need the timestamp t_i and location ls_j . For sensor data, time, location and

sensed value are the minimum components of a sensor measurement tuple [9]. Sensor data streams are produced by various types of sensors that are wirelessly enabled. The streams are often available in real-time and can also be analyzed in real-time. In this project, sensor data streams were stored in a relational database system and analyzed using SQL and Java.

We distinguish several levels of sensor data detail:

- 1) **Level 0:** Level 0 sensor data is raw sensor data that is retrieved from sensor devices. For example, a person's presence can be sensed separately by a motion sensor, audio sensor and light sensor.
- 2) **Level 1:** Level 1 sensor data is aggregated and fused Level 0 sensor data; for example, the raw Level 0 sensor signals are aggregated into the information of a person's presence in room and their trajectories in a building over space and time.
- 3) **Level 2:** Level 2 sensor data is based on Level 1 data, and aggregated into higher-level information about events.

In this project, the input data is Level 1 data, and the objective is to use ontologies to extract Level 2 data.

2.3 Related Work

EasyMeeting [10] is used to organized meetings. It detects bluetooth-enabled equipment of meeting attendants. If someone who is a presenter or key participant is absent, the system plays background music. When everyone has arrived, the system makes the lights dim and show slides. EasyMeeting uses an ontology to express and reason about information. A major difference between our project and EasyMeeting is that we choose the more expressive Common Logic ontology language rather than OWL.

Smart campus [12] is a project for modeling, analyzing and visualizing human movement combined with semantic web in a campus environment. The ontology of smart campus is based on OWL-Time [13] and GeoSPARQL [14]. The ontology captures the terminology related to scheduled events. Smart Campus can show trajectories of student movements by course schedule, which helps people make decisions for course schedules and facility

position. Compared with our project, Smart Campus has a larger scope of relevant events and does not support reasoning.

In [15], an ontology has been created to recognize human activity from video, such as a robbery in a bank. The paper describes several scenarios with an ontology such as shoplifting in a store and an attack in a bank. This work also includes experiments to verify the ontology. Compared with our project, the methods of collecting data are different. Our project uses sensor-based data to collect human activity data, while the project in [15] uses video to acquire data.

Smart Home (SH) has been widely used for assisting elderly people [16]. Smart Homes is a technology based on different kinds of sensors, information and communication technology and ontology. It can support individuals such as the elderly and disabled. By monitoring environmental changes and inhabitant activities, it infers inhabitants' need and can take appropriate actions to support them. So SH can enhance the living quality of elderly people living independently. SH infers real time sensor stream while our project uses stored sensor stream. SH also differs in scope. Our project studies human activity in offices while SH studies elderly people in their own houses.

Chapter 3: ONTOLOGY FOR HUMAN ACTIVITY

3.1 Original Data

The data has been collected as part of a comprehensive study of occupant behavior conducted by Autodesk Research to better simulate office buildings at design time [19]. As Fig. 3.1 shows, it consists of five columns – Occupant (Person ID), Time (Time Stamp indicating the start time), Task, Participants (Number of Participants) and Location (Room). It is not generated from real sensor streams but simulated data, which describes the trajectories of employees in a building. Based on these data, we develop the vocabulary of the ontology.

Occupant	Time	Task	Participants	Location
0	10:49:26	desk_work	1	Office 5016
0	10:53:07	desk_meetin	3	Cubicle 5006
0	11:05:55	desk_meetin	1	Office 5016
0	11:31:45	desk_work	1	Office 5016
0	12:17:35	offsite_break	4	Offsite
0	12:26:37	desk_work	1	Office 5016
0	12:37:38	offsite_break	6	Offsite
0	13:44:33	desk_meetin	2	Cubicle 5011
0	14:46:36	desk_work	1	Office 5016
0	14:47:06	onsite_break	2	Lunch Room
0	15:00:41	desk_work	1	Office 5016
0	15:18:11	washroom_b	1	Washroom (M)
0	15:21:12	desk_work	1	Office 5016
0	15:58:50	desk_meetin	3	Cubicle 5036
0	16:25:30	desk_work	1	Office 5016
0	16:31:08	desk_meetin	2	Cubicle 5006
0	16:39:06	desk_work	1	Office 5016
0	17:15:43	washroom_b	1	Washroom (M)
0	17:21:48	desk_work	1	Office 5016
0	17:34:35	off	1	Offsite
1	10:36:43	desk_work	1	Office 5018
1	10:38:34	desk_meetin	2	Office 5018
1	11:09:00	desk_meetin	2	Office 5018
1	11:36:46	desk_work	1	Office 5018
1	12:05:11	desk_meetin	2	Cubicle 5002
1	12:28:55	offsite_break	3	Offsite

Fig 3.1 Screen shot of Original Data

The layout of the offices is like Fig 3.2.

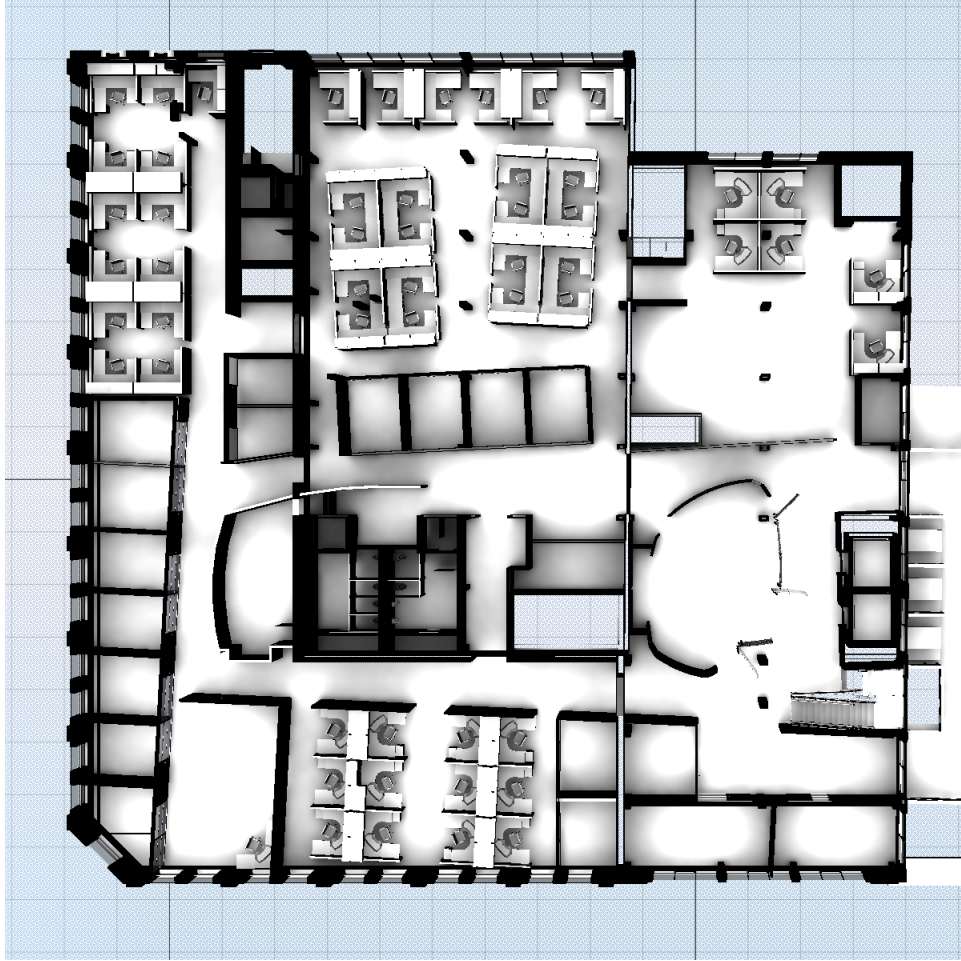


Fig 3.2 Example Space Layout (courtesy of Autodesk Research)

3.2 Domain Analysis

Before creating the ontology, we need to determine the desired functionality of the ontology. Competency questions [17] are used to specify what statements the ontology should be able to infer. They help to tailor the scope of the ontology for the specific application and test when the ontology sufficiently captures the domain. Some competency questions are as below.

1. People from different departments cannot attend the same team meeting.
2. If two spaces are adjacent, their floor numbers are the same.

3. A team meeting always takes place in a room with a projector.
4. A visitor cannot attend a team meeting.
5. A meeting must take place in a cubicle, office or meeting room.
6. Every multiperson event has more than one participant.
7. Every multiperson event cannot have two periods of time.

The first step is about identifying the scope of the domain. The domain of our ontology is to study human behaviors in offices such as meeting, break and work. The ontology has 4 core classes - Space, Person, Activity and TemporalEntity. After defining the domain, we need to decide what vocabulary is necessary for expressing and reasoning about the competency questions..

3.3 Multiple Person Colocation Event

From the original data, we can easily get single person events. But what we really care about are multiple person colocation events. A Multiple person colocation event is in which at least two people are in the same room during the same time period. As the Fig. 5, these four lines represent four single person events in the same room. And these lines are ordered by time. So the black line and green line stand for the first and last single person event. The multiple person colocation event starts when the red line starts (second person enter the room) and ends when the blue line ends (only one person exists in that room).

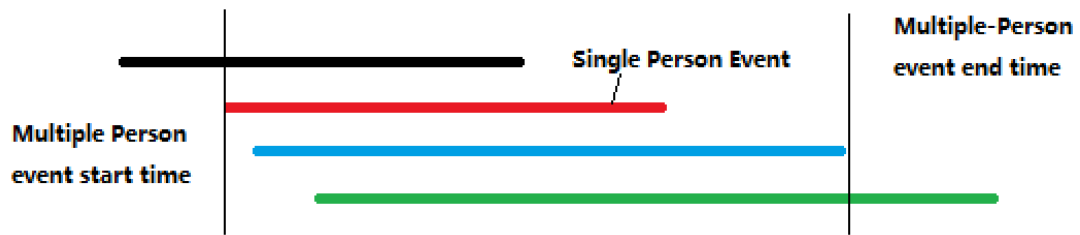


Fig 3.3 Multiple Person Colocation Event Model

3.4 Vocabulary

Classes

Activity

Break

OffsiteBreak

OffsiteBreak is a break when people leave the building.

WaterBreak

WaterBreak is a break when people drink water, coffee or something else.

WashroomBreak

WashroomBreak is a break that happens in washrooms.

Meeting

DeskMeeting

DeskMeeting is a meeting that happens in offices or cubicles.

TeamMeeting

TeamMeeting is a meeting which most people attended come from the same meeting.

Work

DeskWork

DeskWork happens when people work in their cubicles and offices.

Visiting

Visiting happens when people work for visitors

TechWork

TechWork happens when people work in TechRoom.

Equipment

Projector

Printer

Computer

Integer

Person

Employee

Intern

Visitor

Space

BreakSpace

DinningRoom

Washroom

UtilitySpace

Photocopier

TechRoom

WorkSpace

Cubicle

CubicleSpace

CubicleSpace is a space that consists of a set of cubicles.

MeetingRoom

Office

VisitingRoom

3.5 Relationship

Relation Introduction:

1. inside: Instant is in or into the inner part of Interval (Domain: Instant, Range: Interval)
2. take place: Task happens in some place (Domain: Task, Range: Space).
3. participate: People participate in some tasks (Domain: People, Range: Task).
4. adjacent/disjoint: Two locations touch each other (Domain: Space, Range: Space).
5. supervise: Someone observes and directs the work of someone (Domain: Person, Range: Person).
6. begin: The beginning of TemporalEntity is an Instant. (Domain: Instant, Range: TemporalEntity)
7. end: The ending of a TemporalEntity is an Instant. (Domain: Instant, Range: TemporalEntity)
8. last: One Activity lasts for a specific temporal interval. (Domain: Activity, Range: Interval)
9. hasEquipment: A room has a piece of equipment. (Domain: Space, Range: Equipment)
10. contain: One temporal interval includes another interval as subinterval. (Domain: Interval, Range: Interval)
11. hasFloorNumber: Each room has a floor number. (Domain: Space, Range: Integer)
12. PersonInstantRoom: The trajectory relationship between a person, an instant and a room (Parameter 1: Person, Parameter 2: Instant, Parameter 3: Space).

Some Examples of Relations:

take place: Task happens in some place (Domain: Task, Range: Space).

Eg. Sale team meeting will take place on east meeting room this month.

participate: People participate some tasks (Domain: People, Range: Task).

Eg. Manager Tom attends the sale meeting.

adjacent/disjoint: Two locations touch each other (Domain: Space, Range: Space).

Eg. Cubicle 1001 and Office 111 are adjacent.

supervise: Someone observe and direct the work of someone (Domain: Person, Range: Space).

Eg. Manager Tom is the supervisor of Peter who is an intern of IT department

inside: Instant is in or into the inner part of Interval (Domain: Instant, Range: Interval)

Eg. Instant 12:30:00 is inside of Interval from 12:00:00 to 13:00:00.

3.6 Conceptual Model

After domain analysis, we need to build the conceptual model. For better readability, we have split the conceptual model into six submodels described below. The first conceptual model describes the major classes and relationships between them, while the other five describe the subclass relationships for one of the main classes

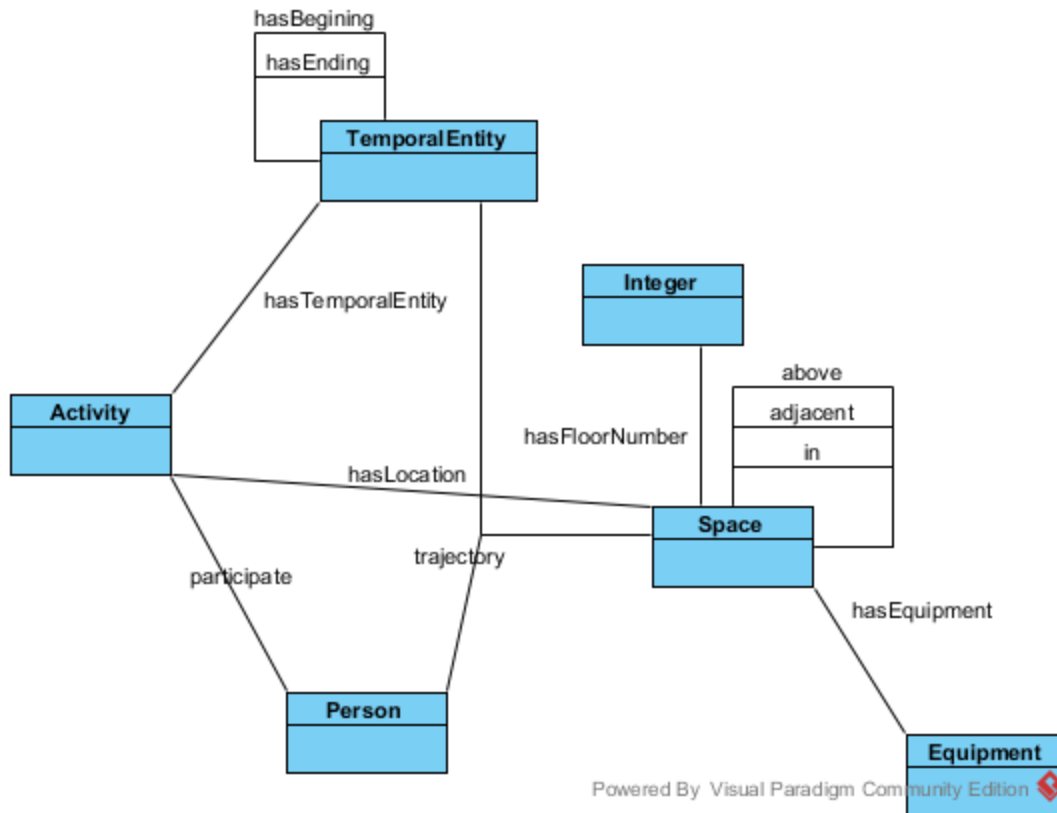


Fig. 3.4 Conceptual Model of Major Classes

As shown in Fig.1, we have 6 major classes - Activity, Person, TemporalEntity, Space, Equipment, and Integer. There are some relationships between them. Only one relation is a ternary relation, that is, trajectory. All other relations are binary. The second conceptual model is about Activity. The model describes the details of the subclass relationships of Activity.

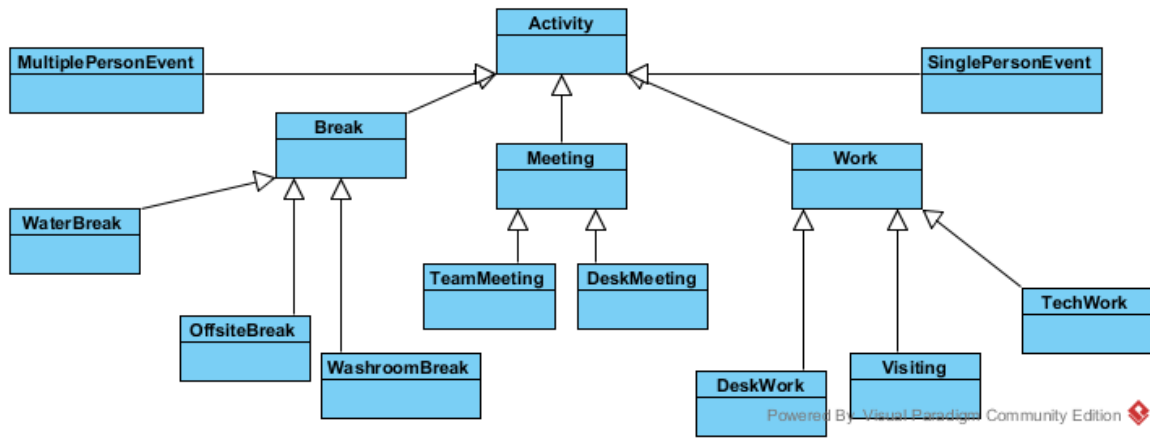


Fig. 3.5 Conceptual Model of Activity

The third conceptual model is about Space. The model describes the subclass relationships of Space.

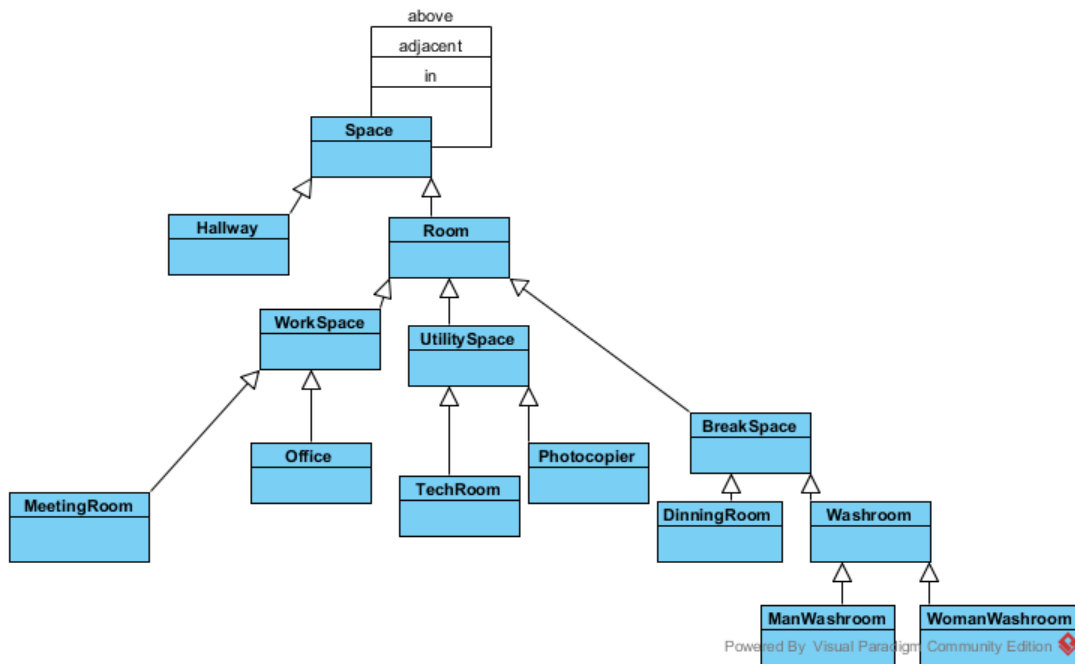


Fig. 3.6 Conceptual Model of Space

The fourth conceptual model is about Person. The model describes the subclass relationships of Person.

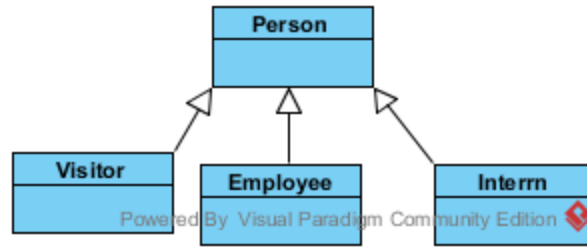


Fig. 3.7 Conceptual Model of Person

The fifth conceptual model is about Time. The model describe the subclass relationships of Temporal Entity.

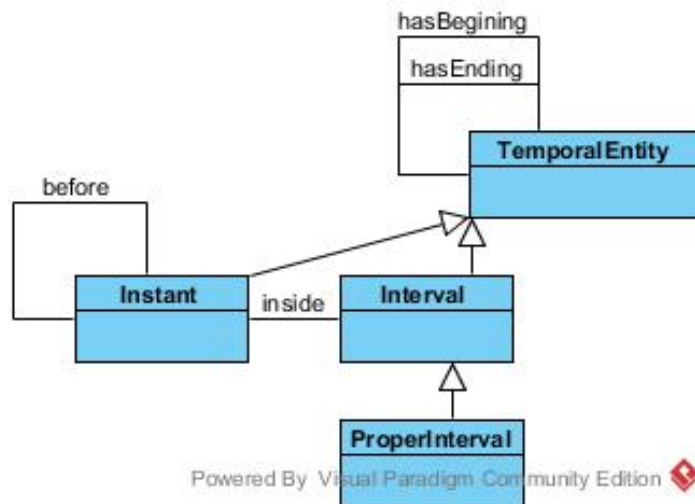


Fig. 3.8 Conceptual Model of Temporal Entity

The sixth conceptual model is about Equipment. The model describes the subclass relationships of Equipment.

3.7 Axioms

There are mainly three kind of axioms – subclass (subsumption) axioms, axioms about time, and axioms characterizing multiple person colocation events. Subclass relationship are about specializations of classes. For example, Team Meeting is the subclass of Meeting. Time axioms are imported from OWL. These axiom are written in Common Logic. For definition axioms, one example is, if more than two persons do something in

the same place during the same period, there exists a colocation event. The complete axioms can be found in Appendix.

The Common Logic axiomatization consists of six modules. The top module is `allRelations`. It imports all other modules and includes all relationships among them. In `allRelations` the major classes – Integer, Equipment, Department, Person, Activity, and TemporalEntity are distinguished. The other five modules are Equipment, Space, Person, Activity, and TemporalEntity.

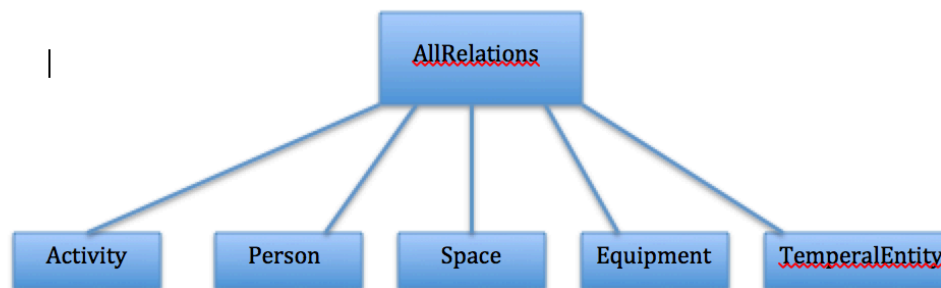


Fig. 3.9 Module Structure

3.7.1 Subsumption Relations

A large part of ontology is about subclass relation. For example, equipment has three subclasses - Projector, Printer and Computer. (A13)

```

(forall (x)
  (iff
    (Meeting x)
    (or
      (DeskMeeting x)
      (TeamMeeting x))))
  
```

Projector is disjoint with printer and computer. (A14)

```

(forall (x)
  (if
  
```

```

        (DeskMeeting x)
      (not
        (TeamMeeting x)
      )
    )
  )
)

```

Other examples of subclass relationships include the different subclasses of Space and of Person. Most importantly, we distinguish several subclasses of Activity – one of them the class Meeting and its two subclasses TeamMeeting and DeskMeeting most relevant to this project.

3.7.2 Time

OWL Time [7] is used to describe the temporal contents. It provides a vocabulary about temporal entities such as instant and interval. Also there are definitions about relations between temporal entities such as before and inside. In our ontology, I reuse part of OWL Time to deal with temporal entity. I reuse part of OWL time and express it in Common Logic. The Common Logic code comes from Colore [17].

One example about time is below. The domain of inside are instants of time and the range are intervals of time. (A55)

```

(forall (t x)
  (if
    (inside t x)
    (and
      (Instant x)
      (Interval x)
    )
  )
)
)

```

Before is a relationship between two instants. (A33)

```

(forall (t1 t2)

```

```

(if
  (before t1 t2)
  (and
    (Instant t1)
    (Instant t2)
  )
)
)

```

Before is not symmetric. (A31)

```

(forall (t1 t2)
  (if
    (before t1 t2)
    (not
      (= t1 t2)
    )
  )
)
)

```

Begins and ends are relationships between instants and temporal entities. (A38)

```

(forall (t x)
  (if
    (begins t x)
    (and
      (Instant t)
      (TemporalEntity x)
    )
  )
)
)

```

Contain is a relationship between proper intervals. If ProperInterval a contains ProperInterval b, the beginning of b is equal to or later than the beginning of a and the ending of b is earlier than or equal to the ending of a. (A46)

```

(forall (a b ta1 ta2 tb1 tb2)
  (if
    (contain a b)
    (exists (ta1 ta2 tb1 tb2)
      (and
        (ProperInterval a)
        (ProperInterval b)
        (not (= a b))
        (begins ta1 a)
        (ends ta2 a)
        (begins tb1 b)
        (ends tb2 b)
        (or
          (before ta1 tb1)
          (= ta1 tb1)
        )
        (or
          (before tb2 ta2)
          (= tb2 ta2)
        )
      )
    )
  )
)

```

3.7.3 Multiple Person Colocation Event

The next axioms express what a Multiple Person Colocation Event is. The first step is to define Single Person Colocation Event. Each single person colocation event has a person, a room, an interval of time. And `SinglePersonEvent` is a subclass of `Activity`. For example, (A19)

```

(forall (x)
  (iff
    (Activity x)
    (or
      (SinglePersonEvent x)
      (MultiplePersonEvent x)
    )
  )
)

```



```

(Work x)
( Break x)
(Meeting x)) ))

```

Another step is to define PersonRoomInstant relation. It is a trajectory relationship among a person, an instant, and a space. (A63)

```

(forall (p l i)
  (if
    (PersonRoomInstant p l i)
    (and
      (Person p)
      (Space l)
      (Instant i)
    )
  )
)

```

Besides the axioms above, we add the axiom that if there is a PersonRoomInstant relationship p, there is a single person event s which has the same person and location and the instant is inside of the interval of s. (A64)

```

(forall (p l i)
  (if
    (PersonRoomInstant p l i)
    (exists (s t)
      (and
        (SinglePersonEvent s)
        (last s t)
        (participate p s)
        (takeplacein s l)
        (inside i t)
      )
    )
  )
)

```

Finally based on the definition of multiple person colocation events (A66)

```

(forall (m l t)
  (if
    (and
      (MultiplePersonEvent m)
      (last m t)
      (takeplacein m l)
    )
    (exists (x y i)
      (and
        (inside i t)
        (not (= x y))
        (PersonRoomInstant x l i)
        (PersonRoomInstant y l i)
      )
    )
  )
)

```

3.7.4 Meeting

Based on the axioms of multiple person colocation event, we can add some axioms to extract different kinds of meetings. If there is a multiple person colocation event, and the location is in a cubicle, office or meeting room, the multiple person colocation event is a meeting. (A25)

```

(forall (m l)
  (if
    (and
      (MultiPersonEvent m)
      (or
        (Cubicle l)
        (Office l)
        (MeetingRoom l)
      )
    )
    (takeplacein m l)
  )
  (Meeting m))

```

If there is a meeting and the location is a meeting room, the meeting is a team meeting.
(A26)

```
(forall (m l)
  (if
    (and
      (MultiPersonEvent m)
      (MeetingRoom l)
      (takeplacein m l)
    )
    (TeamMeeting m)
  )
)
```

If there is a meeting and the location is a cubicle or office, the meeting is a desk meeting.
(A27)

```
(forall (m l)
  (if
    (and
      (MultiPersonEvent m)
      (or
        (Cubicle l)
        (Office l)
      )
      (takeplacein m l)
    )
    (DeskMeeting m)
  )
)
```

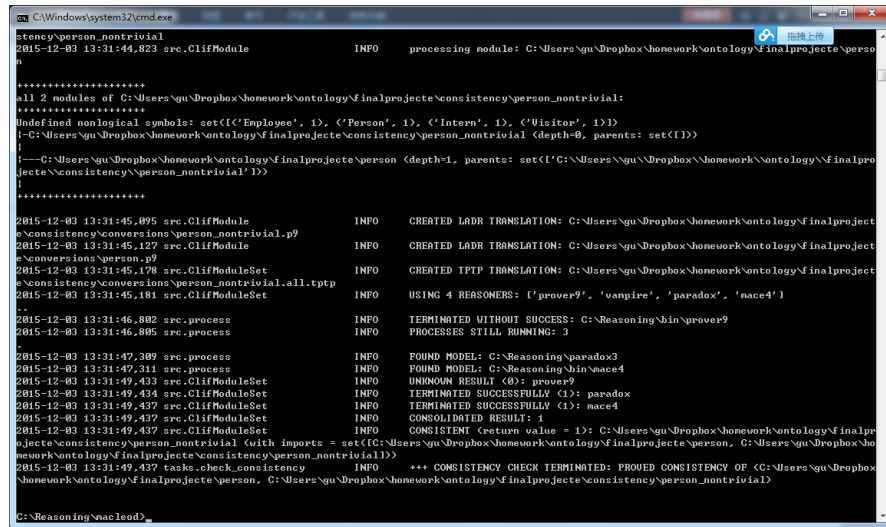
3.8 Ontology Evaluation

Ontology verification [18] is the process of ensuring that the ontology is both logically correct and describes the domain correctly. For the first part, we check the consistency of the ontology. For the second part, a set of competency questions are used [11].

3.8.2 Consistency

Consistency means this ontology does not contain any contradiction. If the ontology is not consistent, it can prove arbitrary statements, rendering it useless. Therefore, proving consistency is an important step in the verification of the ontology.

The ontology is specified with common logic. After creating ontology, we use Macleod [17] to test the ontology for consistency. Macleod is a software which currently utilizes four reasoners in parallel – Vampire, Prover9, Mac4 and Paradox. For our whole ontology, paradox gets a model which can be found in Appendix (consistence result). It means the whole ontology $O = \{A1, A2, \dots, A104\}$ is consistent.



```
C:\Windows\system32\cmd.exe
consistency\person_nontrivial
2015-12-03 13:31:44,823 src.ClifModule INFO processing module: C:\Users\gu\Dropbox\homework\ontology\finalproject\perso
n
*****
all 2 modules of C:\Users\gu\Dropbox\homework\ontology\finalproject\consistency\person_nontrivial:
*****
Undefined nonlogical symbols: set{(('Employee', 1), ('Person', 1), ('Intern', 1), ('Visitor', 1))
!- C:\Users\gu\Dropbox\homework\ontology\finalproject\consistency\person_nontrivial (depth=0, parents: set{()})
!- C:\Users\gu\Dropbox\homework\ontology\finalproject\person (depth=1, parents: set{(C:\Users\gu\Dropbox\homework\ontology\finalpro
ject\consistency\person_nontrivial' 1)})
*****
2015-12-03 13:31:45,095 src.ClifModule INFO CREATED LADR TRANSLATION: C:\Users\gu\Dropbox\homework\ontology\finalproject
e\consistency\conversions\person_nontrivial.p9
2015-12-03 13:31:45,127 src.ClifModule INFO CREATED LADR TRANSLATION: C:\Users\gu\Dropbox\homework\ontology\finalproject
e\conversions\person.p9
2015-12-03 13:31:45,178 src.ClifModuleSet INFO CREATED TPTP TRANSLATION: C:\Users\gu\Dropbox\homework\ontology\finalproject
e\consistency\conversions\person_nontrivial.all.tptp
2015-12-03 13:31:45,181 src.ClifModuleSet INFO USING 4 REASONERS: ['prover9', 'vampire', 'paradox', 'mac4']
2015-12-03 13:31:46,802 src.process INFO TERMINATED WITHOUT SUCCESS: C:\Reasoning\bin\prover9
2015-12-03 13:31:46,805 src.process INFO PROCESSES STILL RUNNING: 3
2015-12-03 13:31:47,309 src.process INFO FOUND MODEL: C:\Reasoning\bin\paradox3
2015-12-03 13:31:47,311 src.process INFO FOUND MODEL: C:\Reasoning\bin\mac4
2015-12-03 13:31:49,433 src.ClifModuleSet INFO UNKNOWN RESULT (0): prover9
2015-12-03 13:31:49,434 src.ClifModuleSet INFO TERMINATED SUCCESSFULLY (1): paradox
2015-12-03 13:31:49,437 src.ClifModuleSet INFO TERMINATED SUCCESSFULLY (1): mac4
2015-12-03 13:31:49,437 src.ClifModuleSet INFO CONSOLIDATED RESULT: 1
2015-12-03 13:31:49,437 src.ClifModuleSet INFO CONSISTENT (return value = 1): C:\Users\gu\Dropbox\homework\ontology\finalpr
oject\consistency\person_nontrivial (with imports = set{(C:\Users\gu\Dropbox\homework\ontology\finalproject\person, C:\Users\gu\Dropbox\ho
mework\ontology\finalproject\consistency\person_nontrivial)})
2015-12-03 13:31:49,437 tasks.check_consistency INFO *** CONSISTENCY CHECK TERMINATED: PROVED CONSISTENCY OF (C:\Users\gu\Dropbox\
homework\ontology\finalproject\person, C:\Users\gu\Dropbox\homework\ontology\finalproject\consistency\person_nontrivial)
C:\Reasoning\macleod>
```

Fig 3.10 Output of Macleod for the Consistency Test

3.8.1 Competency

Competency questions are a set of logical statements that characterize what is expected from a new ontology and drive the development of new ontology [11].

The development of the ontology was guided by the set of competency questions below. They were checked against the ontology using the Macleod toolset from [17] to test whether they can be inferred from the ontology. The results of competency questions are as below. None of the competency questions can be disproven, which would mean that the ontology is insufficiently axiomatized. Though for some of the competency questions, we could not find a proof either, requiring further tests in future.

Question 1: People from different departments cannot attend the same team meeting.

The common logic expression of the question is as below. This question can be proven from the ontology. The proof is in folder consistency folder1.

```
( forall (x y z a b)
  (if
    (and
      (hasdepartment x a)
      (hasdepartment y b)
      (not (= a b))
      (not (= x y))
      (TeamMeeting z)
    )
    (not
      (and
        (participate x z)
        (participate y z)
      )
    )
  )
)
```

Question 2: If two spaces are adjacent, their floor numbers are same.

The common logic expression of the question is as below. This question can be proven. The proof is in folder consistency 2.

```

(forall (a b x y)
  (if
    (and
      (adjacent a b)
      (hasFloor x a)
      (hasFloor y b)
      (not (= a b))
    )
    (= a b))
  )
)

```

Question 3: A team meeting always takes place in a room with a projector.

The common logic expression of the question is as below. This question cannot be proven. Macleod shows no result, that is, neither consistency nor inconsistency can be proven. The Macleod result can be found in folder consistency3.

```

(forall (a b e)
  (if
    (and
      (TeamMeeting a)
      (Projector e)
      (takeplacein a b)
    )
    (hasEquipment b e)
  ))

```

Question 4: A visitor cannot attend a team meeting.

The common logic expression of the question is as below. This question cannot be proven either The Macleod result can be found in folder consistency4.

```

(not
  (exists (a m)
    (and
      (Visitor a)
      (TeamMeeting m)
      (participate a m)
    )
  )
)

```

```
)
)
```

Question 5: A meeting must take place in cubicles, offices and meeting room.

The common logic expression of the question is as below. This question cannot be proven either. The Macleod result can be found in folder consistency5.

```
(forall (m l)
  (if
    (and
      (Meeting m)
      (takeplacein m l)
    )
    (or
      (Cubicle l)
      (Office l)
      (MeetingRoom l)
    )
  )
)
```

Question 6: All multiple person colocation event has more than one participant.

The common logic expression of the question is as below. This question can be proven.

The proof is in folder consistency 6.

```
(forall (m)
  (if
    (MultiPersonEvent m)
    (exists (a b)
      (and
        (participate a m)
        (participate b m)
        (not (= a b))
      )
    )
  )
)
```

)

Question 7: All multiple person colocation event cannot have two periods of time.

The common logic expression of the question is as below. This question can be proven.

The proof is in folder consistency 7.

```
(forall (m a b)
  (and
    (MultiPersonEvent m)
    (last m a)
    (last m b)
  )
  (= a b)
)
```


Chapter 4: DATA ANALYSIS AND ALGORITHMS

In this section, we present several algorithms to extract the ontological concepts presented in Chapter 4 based on Level 1 sensor data.

4.1 Single Person Colocation Event

Fig. 4.1 depicts the structure and type of the Level 1 data; a tuple consists of the *Occupant_id*, *Time* and *Location* attributes.

Occupant	Time	Location
0	10:49:26	Office 5016
0	10:53:07	Cubicle 5006
0	11:05:55	Office 5016
0	11:31:45	Office 5016
0	12:17:35	Offsite
0	12:26:37	Office 5016
0	12:37:38	Offsite
0	13:44:33	Cubicle 5011
0	14:46:36	Office 5016
0	14:47:06	Lunch Room
0	15:00:41	Office 5016
0	15:18:11	Washroom (M)
0	15:21:12	Office 5016
0	15:58:50	Cubicle 5036
0	16:25:30	Office 5016
0	16:31:08	Cubicle 5006
0	16:39:06	Office 5016
0	17:15:43	Washroom (M)
0	17:21:48	Office 5016
0	17:34:35	Offsite
1	10:36:43	Office 5018
1	10:38:34	Office 5018
1	11:09:00	Office 5018
1	11:36:46	Office 5018

Fig. 4.1 Screen shot of Trajectory data

Based on Fig. 4.1, the following SQL query transforms the data into new tuples that represent how long a person is present in a single location.

Select OccupantId, starttime, Location from `People Movement`

If we order the table by *Occupant_ID* and *Time*, each following tuple represents a person starting to be present in a new location, and this signals the end of the person's presence in the previous location;. In this way, we create a new table that contains single person event data as Fig. 4.2.

OccupantId	starttime	endtime	duration	location
0	10:49:26	10:53:07	00:03:41	Office 5016
0	10:53:07	11:05:55	00:12:48	Cubicle 5006
0	11:05:55	11:31:45	00:25:50	Office 5016
0	11:31:45	12:17:35	00:45:50	Office 5016
0	12:17:35	12:26:37	00:09:02	Offsite
0	12:26:37	12:37:38	00:11:01	Office 5016
0	12:37:38	13:44:33	01:06:55	Offsite
0	13:44:33	14:46:36	01:02:03	Cubicle 5011
0	14:46:36	14:47:06	00:00:30	Office 5016
0	14:47:06	15:00:41	00:13:35	Lunch Room
0	15:00:41	15:18:11	00:17:30	Office 5016
0	15:18:11	15:21:12	00:03:01	Washroom (M)
0	15:21:12	15:58:50	00:37:38	Office 5016
0	15:58:50	16:25:30	00:26:40	Cubicle 5036
0	16:25:30	16:31:08	00:05:38	Office 5016
0	16:31:08	16:39:06	00:07:58	Cubicle 5006
0	16:39:06	17:15:43	00:36:37	Office 5016
0	17:15:43	17:21:48	00:06:05	Washroom (M)
0	17:21:48	17:34:35	00:12:47	Office 5016

Fig. 4.2 Screen Shot of Single Person Event

4.2 2-person Colocation Events

To derive multiple person colocation events, computing 2-person colocation events is the next, intermediate step. The definition of 2-person colocation event is as Fig. 4.3.

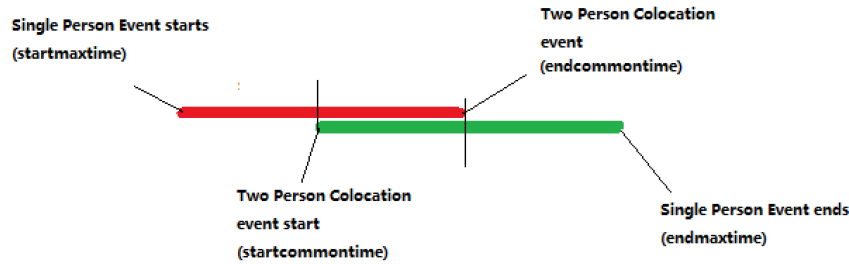


Fig. 4.3 2-person Colocation Event Model

Based on Fig. 4.3, we can compute a 2-person colocation event by a simple loop function. From Fig. 4.4, the data has five attribute – *Occupant_id1*, *Occupant_id2*, *startcommontime* (the beginning of common time), *endcommontime* (the ending of common time) and location.

Occupant_id1	Occupant_id2	startcommontime	endcommontime	location
1	0	12:37:38	12:51:44	Offsite
1	2	13:16:16	13:28:56	Lunch Room
3	0	12:17:35	12:26:37	Offsite
3	0	12:37:38	12:51:44	Offsite
3	1	12:28:55	12:51:44	Offsite
3	4	16:33:37	17:18:34	Office 5021
1	4	18:05:01	18:09:20	Office 5018
4	5	10:12:54	10:13:03	Washroom (W)
4	5	10:28:31	10:31:11	Office 5021
0	5	13:36:37	13:43:47	Offsite
2	6	10:54:36	11:13:55	Office 5049
0	6	12:17:35	12:26:37	Offsite
3	6	12:17:35	12:26:37	Offsite
7	2	10:53:20	10:54:36	Office 5050
2	7	11:50:36	11:59:14	Office 5019
6	7	16:01:49	16:06:43	Office 5050
6	7	16:07:06	16:08:24	Office 5050
0	8	12:19:21	12:26:37	Offsite

Fig. 4.4 Screen shot of 2-person Colocation

This results are generated by a MySQL script which can be found in the Appendix (Script of generating 2-person colocation event).

4.3 Multiple Person Colocation Events

According to the definition of Colocation events, at least two people are present at any time at the same place. We aggregate 2-person colocation events to get multiple person colocation events. Each multiple colocation event has an Id and two or more participant ids.. In Fig. 4.5, ten 2-person colocation events are depicted. We can see the first 2-*person colocation event id*(452) with the two *Participant_id*(23, 46). Similar to Fig. 4.4, these 2-person colocation events are ordered by start time. That is, 452 is the first 2-person colocation event for a certain room. Event 359 is the last 2-person colocation event in this room for the given dataset.

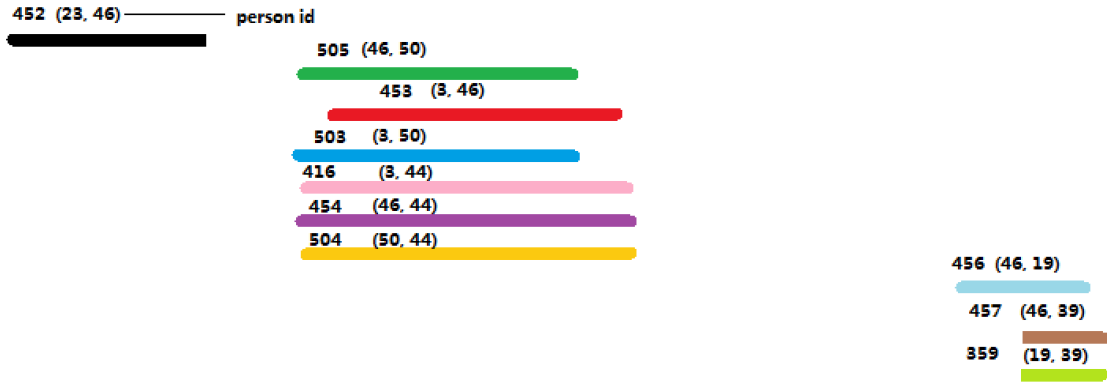


Fig. 4.5 2-person Colocation Events in Cubicle 5056

The algorithm of getting multiple person colocation events is described below.

- 1) The first step is to find which 2-person colocation events overlap with the 2-person colocation event 452. Obviously, there is no other event.
- 2) Find which 2-person colocation events overlap with 505. Obviously, there are five other 2-colocation events. Because these 2-person colocation events are ordered by start time. The first next 2-person colocation event is 453.

Based on our algorithm(*at_least_two_persons*), if endcommontime of 505 is later than the startcommontime of 453, 505 merges with 453. 453 is deleted. Occupants of 505 merges with the occupants of 453 is (46, 50, 3). Then we get Fig. 4.6.

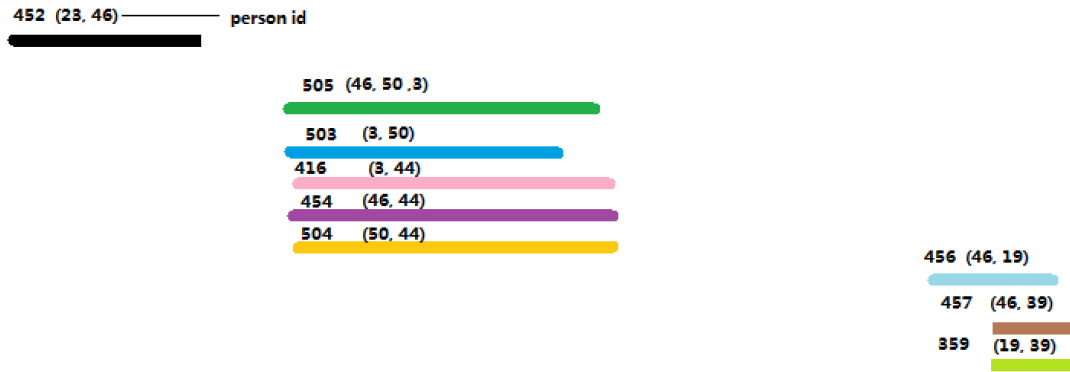


Fig. 4.6 505 merges 453

Then we continue to find that 505 overlaps with the events 503, 416, 454, and 504. So 505 merges with them. As result of the algorithm, we get Fig. 4.7.

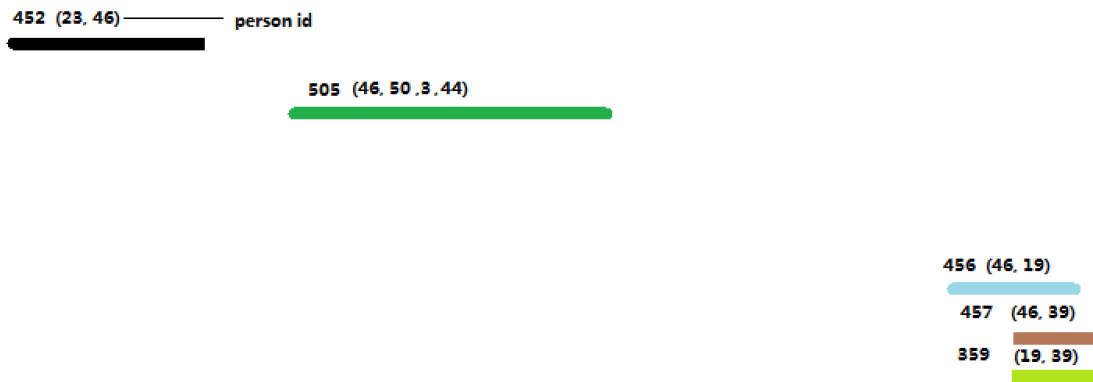


Fig. 4.7 505 merge 503, 416, 454 and 504

After event 505, we need to find which event overlaps event 456.

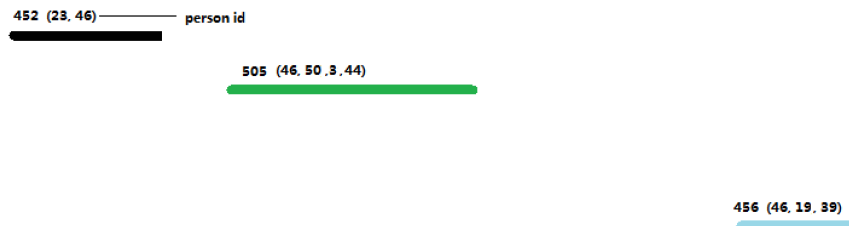


Fig. 4.8 event 456 merges with event 457 and 359.

Finally we get three multiple person events for the location:

(start_time: 10:18:34 end_time: 10:42:12 participants: 23 46)

(start_time: 11:43:39 end_time: 12:15:16 participants: 46 50 3 44)

(start_time: 18:07:27 end_time: 19:22:48 participants: 46 19 39)

4.4 Pseudo Code of Generating Multiple Person Colocation Event

After we compute all 2-person colocation events, we need to divide them by location and order them by time. The input to the algorithm are all 2-person colocation events in the same room.

Table 1: The Algorithm of Generate Multiple Person Colocation Events

Variables	Descriptions
cl	Collection of 2-Person Colocation Events
copyCl	get it by cloning collection of 2-Person Colocation Events
<pre>//Aggregate 2-person colocation events into multiple-person colocation events input cl input cl get copyCl by cloning cl for each element of cl set root to the element while root is not in copyCl if root is the last element of cl break the while loop else set root to the next element of root endif endwhile for each element of copyCl set node to the element if root.id is not equal to node.id and root.starttime is before node.starttime and node.starttime is before root.endtime root add node as its child root.endtime is the later one between root.endtime and node.endtime copyCl remove element node endif endfor endfor</pre>	

4.5 Meeting Classification

Using the axioms from the application ontology from Chapter 3, we can extract more useful information. In this project, the next step is the classification of meeting types. Based on the data and ontology, meetings are classified based on location. For example, collocation events that take place in a kitchen are not considered ‘meetings’, but other locations such as *meeting room* or *desk* help to classify meetings further.

Based on the definition of meeting, this classification can be easily extracted by a SQL query.

```
SELECT * FROM building_ontology.ColocationEvent where location like '%Cubicle%'
or location like '%Office%' or location like '%Meeting%'
```

Similarly, we can use such SQL scripts to get desk meetings and team meetings.

Desk Meeting

```
SELECT * FROM building_ontology.ColocationEvent where location like '%Cubicle%'
or location like '%Office%'
```

Team Meeting

```
SELECT * FROM building_ontology.ColocationEvent where location like
"%Meeting%" or location like "%RD%";
```

The detail information of four team meetings are in Table 2

Table 2 Detail Information of Team Meetings

start_time	end_time	location	participant
14:41:35	15:12:47	East Meeting Room	30 57 17
08:31:00	09:15:41	RD Team Room	56 34
14:06:16	15:09:31	RD Team Room	1 58
15:23:21	16:13:40	West Meeting Room	18 14 39 16 17 19 12 13 9 41 36 51

After running these scripts, we extract 160 meetings, and they are classified into 156 desk meetings and 4 team meetings. The average time of Desk Meeting is 5 mins, which is much shorter than team meeting.

Table 3 Meeting Summary

Meeting Type	Desk Meeting	Team Meeting
Number	156	4
Average Time	5 mins	47 mins

4.6 Verification

At last, we need to test whether all derived meetings are consistent with the information in the original data. Clearly, single person events are generated by the original data directly. Each single person event is generated by one row and the next row of original data if two rows have the same occupant id. Therefore, we just need to test whether meetings are consistent with single person events in the original dataset. There are two directions to test by Java Scripts.

Mapping from meeting to single person events

After generating meetings, we check each participant of that meeting. If the periods of time of all participants in that meeting location overlap with single person events, the meeting is consistent with original data.

Combined with the location and time interval of a meeting, participants of the meeting can be regarded as (person location, time interval). For example a meeting of our meeting results.

(start: 13:27:55, end: 13:44:33, location: Cubicle 5000 , participant: 11 22)

It can be divided into two triple relationships.

(start: 13:27:55, end: 13:44:33, location: Cubicle 5000 , participant: 11)

(start: 13:27:55, end: 13:44:33, location: Cubicle 5000 , participant: 22)

If we can find two single person events, which have the same location and participant id and their time intervals overlap, it means the meeting is consistent with original data.

Easily we find two single person events

(start: 13:27:55, end: 13:44:33, location: Cubicle 5000 , participant: 11)

(start: 13:27:55, end: 13:44:33, location: Cubicle 5000 , participant: 22)

So the meeting is consistent with original data. In the same way, we try every meeting automatically by scripts, all meeting are correct and consistent with original data.

From single person event to meeting

In the opposite direction, the input is every single person event, that is related to a meeting. If one single person event has the same location and the person id is also included in the participants ids of the meeting and their time intervals overlap, this single person event entry is validated and consistent with meeting results.

There are 404 related single person event, after running scripts, 342 of them are correct, the percentage is 85% which is good since the original data sets contains some inconsistencies. One of the reasons is the noise of the original data that was created purposely. As Fig. 4.9, there is only one participant in a meeting, which contradicts the axioms of the concept “meeting” that every meeting has at least two participants.

id	OccupantId	starttime	Task	Participates	Location
3	0	11:05:55	desk_meeting	1	Office 5016
31	1	15:13:10	desk_meeting	1	Office 5018
33	1	15:26:02	desk_meeting	1	Office 5018
41	1	17:28:15	desk_meeting	1	Office 5018
48	1	21:34:34	desk_meeting	1	Office 5018
66	2	16:35:23	desk_meeting	1	Office 5019
94	4	11:34:36	desk_meeting	1	Office 5021
96	4	12:55:15	desk_meeting	1	Office 5021
100	4	13:56:41	desk_meeting	1	Office 5021
112	4	17:37:34	desk_meeting	1	Office 5021
148	6	09:44:05	desk_meeting	1	Office 5049
157	6	11:40:18	desk_meeting	1	Office 5049
177	6	16:40:20	desk_meeting	1	Office 5049
192	7	09:35:51	desk_meeting	1	Office 5050
225	8	10:02:16	desk_meeting	1	Office 5052
231	8	12:02:22	desk_meeting	1	Office 5052
249	9	09:35:46	desk_meeting	1	Office 5066
265	9	13:53:57	desk_meeting	1	Office 5066

Fig. 4.9 noise in original data.

Chapter 5: **CONCLUSIONS AND FUTURE WORK**

In this project, we built an application ontology for human activity in offices. We constructed the ontology using Common Logic. The ontology was tested with Macleod and it is consistent. Then, we can translate the ontology into Java algorithms. With these algorithms, we can use the ontology to reason about data and extract meeting information in that building from sensor data (presence data of people in locations). Based on the original data, we tested and verified the ontology and the developed algorithms. We found all meetings are consistent with the original data set. About 85% of the related original data is consistent with our extracted meetings. One of the reasons of the discrepancy is purposely generated noise in the original data. Above all, the result is good.

In the future, we will use real raw sensor data, potentially real-time sensor stream to recognize human activity based on the developed ontology. One of drawback of the project is lack of data. The original data has about 1500 entries, which are entries for a single day of movement. Once we have larger amounts of data, we will use Hadoop and Spark to rewrite the Java algorithms as stream based algorithms. Further, we need to classify meetings not simply by location but based on more conditions, for example, by core common time and equipment. Core common time is the period of time when all participants are in a room. For equipment, we can add an axiom – if it is a team meeting, the projector is on. Another improvement could be that the ontology can recognize not only meeting but also break or other activities.

BIBLIOGRAPHY

- [1] Hoes, P., et al. "User behavior in whole building simulation." *Energy and buildings* 41.3 (2009): 295-302.
- [2] Rodriguez, Natalia Daz, et al. "A fuzzy ontology for semantic modelling and recognition of human behaviour." *Knowledge-Based Systems* 66 (2014): 46-60.
- [3] Wikipedia contributors. "Knowledge-based systems." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 27 Aug. 2015. Web. 29 Sep. 2015.
- [4] Menzel, Christopher. "Reference Ontologies-Application Ontologies: Either/Or or Both/And?." *KI Workshop on Reference Ontologies and Application Ontologies*. 2003
- [5] Gruber, Thomas "A translation approach to portable ontology specifications." *Knowledge acquisition* 5.2 (1993): 199-220.
- [6] Wikipedia contributors. "Ontology (information science)." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 20 Nov. 2015. Web. 7 Dec. 2015.
- [7] International Electrotechnical Commission (ISO/IEC) ISO 24707:2007 Common Logic(CL), [http://standards.iso.org/ittf/PubliclyAvailableStandards/c039175\ISO\IEC_24707_2007\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c039175\ISO\IEC_24707_2007(E).zip), October 2007
- [8] OWL Working Group. "Web Ontology Language (OWL)." N.p., 11 Dec. 2012. Web.
- [9] S. Nittel, 2015, "Real-time Sensor Data Streams", *SIGSPATIAL Newsletter*, Special Issue "Geosensor Networks", Vol 7(2), pp. 22-28.
- [10] Fox, Armando, et al. "Integrating information appliances into an interactive workspace." *Computer graphics and Applications*, IEEE 20.3 (2000): 54-65.
- [11] Grüninger, M. & Mark S. Fox "The Role of Competency Questions in Enterprise Engineering", *IFIP WG5.7 Workshop on Benchmarking - Theory and Practice*, Trondheim, Norway, 1994
- [12] Fan, Junchuan, and Kathleen Stewart. "An Ontology-based Framework for Modeling Movement on a Smart Campus."

- [13] McGuinness, Deborah L., and Frank Van Harmelen. "OWL Web Ontology Language Overview." N.p., 10 Feb. 2004. Web.
- [14] GeoSPARQL <http://www.opengeospatial.org/standards/geosparql>
- [15] Akdemir, Umut, Pavan Turaga, and Rama Chellappa. "An ontology based approach for activity recognition from video." Proceedings of the 16th ACM international conference on Multimedia. ACM, 2008.
- [16] Okeyo, George, Liming Chen, and Hui Wang. "Combining ontological and temporal formalisms for composite activity modelling and recognition in smart homes." Future Generation Computer Systems 39 (2014): 29-43.
- [17] Hahmann, Torsten. "A Reconciliation of Logical Representations of Space: from Multidimensional Mereotopology to Geometry" *Univ. of Toronto, Dept. of Comp. Science*, 2013, Macleod software available from <https://github.com/thahmann/macleod>
- [18] Grüninger, Michael, Torsten Hahmann, T, Ali Hashemi, and Darren Ong. "Ontology verification with repositories." *Conf. on Formal Ontology in Inf. Systems (FOIS-10)*, IOS Press, 2010, 317-330.
- [19] Goldstein, Rhys, Alex Tessier, and Azam Khan. "Schedule-Calibrated Occupant Behavior Simulation." In Proceedings of the Symposium on Simulation for Architecture and Urban Design, Orlando, USA, 2010.

Appendix

Script of generating 2-person colocation event

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `curdemo7`()
BEGIN
DECLARE done boolean DEFAULT FALSE;
DECLARE l varchar(100);
DECLARE lo varchar(100);
DECLARE Occupant_id INT;
declare p1 INT;
DECLARE Occupant_ido INT;
DECLARE start_time, end_time Time;
DECLARE start_timeo, end_timeo Time;
DECLARE cur1 CURSOR FOR
SELECT OccupantId,starttime,endtime, location FROM PeopleDuration;
DECLARE cur2 CURSOR FOR
SELECT OccupantId,starttime,endtime, location FROM PeopleDuration;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
OPEN cur1;
read_loop: LOOP
SET p1 = 0;
FETCH cur1 INTO Occupant_id, start_time, end_time, l;
open cur2;
reado_loop: LOOP
FETCH cur2 INTO Occupant_ido, start_timeo, end_timeo, lo;
if lo = l and Occupant_ido < Occupant_id
and start_timeo < end_time and start_timeo > start_time
and end_time < end_timeo then
INSERT INTO ColocationEvent2_modified
(occupant_id1, occupant1_start, occupant1_end,
occupant_id2, occupant2_start, occupant2_end,location,
startmaxtime, endmaxtime, startcommontime, endcommontime)
VALUES(Occupant_id,start_time, end_time,Occupant_ido,
start_timeo, end_timeo,lo, start_time end_timeo, start_timeo, end_time);
elseif lo = l and Occupant_ido < Occupant_id
and start_time > start_timeo and end_timeo > start_time
and end_timeo < end_time then
INSERT INTO ColocationEvent2_modified
(occupant_id1, occupant1_start, occupant1_end,
occupant_id2, occupant2_start, occupant2_end,location,
startmaxtime, endmaxtime, startcommontime, endcommontime)
VALUES(Occupant_ido,start_timeo, end_timeo,
Occupant_id, start_time, end_time,lo,
start_timeo, end_time, start_time, end_timeo);
elseif lo = l and Occupant_ido < Occupant_id
and start_time >= start_timeo
and end_time <= end_timeo then
INSERT INTO ColocationEvent2_modified
(occupant_id1, occupant1_start, occupant1_end,
occupant_id2, occupant2_start, occupant2_end,location,
startmaxtime, endmaxtime, startcommontime, endcommontime)
VALUES(Occupant_ido,start_timeo, end_timeo,Occupant_id,
start_time, end_time,lo, start_timeo,
end_timeo, start_time, end_time);
```

```

elseif lo = 1 and Occupant_ido < Occupant_id
  and start_time <= start_timeo
  and end_time >= end_timeo then
  INSERT INTO ColocationEvent2_modified
  (occupant_id1, occupant1_start, occupant1_end, occupant_id2,
  occupant2_start, occupant2_end, location, startmaxtime,
  endmaxtime, startcommontime, endcommontime)
  VALUES(Occupant_id, start_time, end_time, Occupant_ido,
  start_timeo, end_timeo, lo, start_time,
  end_time, start_timeo, end_timeo);
end if;
  IF Occupant_ido < 58 THEN
    ITERATE reado_loop;
  END IF;
  LEAVE reado_loop;
END LOOP;
  close cur2;
IF done THEN
  LEAVE read_loop;
END IF;
END LOOP;
CLOSE cur1;
END

```

Script of generating multiple person colocation event

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Iterator;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
public class Tree {
    public static ArrayList<Node> location = new ArrayList<Node>();
    private static int recursion = 500;
    public static void readCSV() {
        String csvFile = "2_person_colocation.csv";
        BufferedReader br = null;
        String line = "";
        String cvsSplitBy = ",";
        ArrayList<String> ls = new ArrayList<String>();
        Node locationNode = null;
        ArrayList<Node> al = new ArrayList<Node>();
        try {
            br = new BufferedReader(new FileReader(csvFile));
            line = br.readLine();
            while ((line = br.readLine()) != null) {
                String[] entry = line.split(cvsSplitBy);
                if(!ls.contains(entry[7])){
                    System.out.println(entry[7]);
                    ls.add(entry[7]);
                    locationNode = new Node(0, "", "", entry[7], al, null);
                    al = new ArrayList<Node>();
                    locationNode.setChildren(al);
                    location.add(locationNode);
                    System.out.println(location.size());
                }
                ArrayList<Integer> participants = new ArrayList<Integer>();
                participants.add(Integer.parseInt(entry[1]));
                participants.add(Integer.parseInt(entry[4]));
                Node node = new Node(Integer.parseInt(entry[0]),
                    entry[10], entry[11], entry[7], null, participants );
                al.add(node);
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (br != null) {
                try {
                    br.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```



```

}

public static ArrayList dbConnection(){
    String driver = "com.mysql.jdbc.Driver";
    String dbName = "building_ontology";
    String passwrod = "root";
    String userName = "root";
    String url = "jdbc:mysql://127.0.0.1:8889/" + "building_ontology";
    String sql = "select * from ColocationEvent2_modified "
        + "where location not in "
        + "('West Meeting Room', 'Offsite', 'Kitchen', "
        + "'Lunch Room', 'Washroom (W)', 'Washroom (M)' )"
        + "Order By location, startcommontime asc";

    ArrayList<String> ls = new ArrayList<String>();
    Node locationNode = null;
    ArrayList<Node> al = new ArrayList<Node>();
    try {
        Class.forName(driver);
        Connection conn = DriverManager.getConnection(url, userName,
            passwrod);
        PreparedStatement ps = conn.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();
        while (rs.next()) {
            if(!ls.contains(rs.getString(8))){
                System.out.println(rs.getString(8));
                ls.add(rs.getString(8));
                locationNode = new Node(0, sql, sql,rs.getString(8),
                    al, null);
                al = new ArrayList<Node>();
                locationNode.setChildren(al);
                location.add(locationNode);
                System.out.println(location.size());
            }
            ArrayList<Integer> participants = new ArrayList<Integer>();
            participants.add(rs.getInt(2));
            participants.add(rs.getInt(5));
            Node node = new Node(rs.getInt(1), rs.getString(11),
                rs.getString(12), rs.getString(8), null, participants );
            System.out.println(node.getId() + " " + node);
            al.add(node);
        }
        if (rs != null) {
            try {
                rs.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if (ps != null) {
            try {
                ps.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if (conn != null) {
            try {
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

    }
}
} catch (Exception e) {
    e.printStackTrace();
}
return location.get(0).getChildren();
}

public static void atLeasttwo(ArrayList<Node> al){
    ArrayList<Node> copyAl = new ArrayList<Node>();
    boolean lastElement = false;
    for(Node node: al){
        copyAl.add(node);
    }
    for(Iterator<Node> alIterator = al.iterator();
        alIterator.hasNext();){
        Node root = alIterator.next();
        while(!copyAl.contains(root)){
            if(!alIterator.hasNext()){
                lastElement = true;
                break;
            }
            else{
                root = alIterator.next();
            }
        }
    }
    for(Iterator<Node> copyAlIterator = copyAl.iterator();
        copyAlIterator.hasNext();){
        Node node = copyAlIterator.next();
        boolean childrenEqual = false;
        if(node.getStartTime() == root.getStartTime()
            && node.getEndTime() == root.getEndTime() &&
            node.getParticipants().containsAll(root.getParticipants())
            && root.getParticipants().containsAll(node.getParticipants())){
            childrenEqual = true;
        }
        if(!lastElement && root.getId() != node.getId()
            && root.getStartTime().compareTo(node.getStartTime()) <= 0 &&
            root.getEndTime().compareTo(node.getStartTime()) > 0
            && !childrenEqual){
            String endTime = root.getEndTime().compareTo(node.getEndTime())>0 ?
            root.getEndTime() : node.getEndTime();
            root.setEndTime(endTime);
            for(Integer rp: root.getParticipants()){
                for(Iterator<Integer> np = node.getParticipants().iterator();
                    np.hasNext();){
                    if((int)rp == (int)np.next()){
                        np.remove();
                    }
                }
            }
            root.getParticipants().addAll(node.getParticipants());
            String participants = new String("");
            for(Integer np: root.getParticipants()){
                String block = " "+np;
                participants = participants + block;
            }
            copyAlIterator.remove();
        }
    }
}

```

```

        }
    }
    for(Node node: copyAl){
        System.out.println("id: "+node.getId());
        System.out.println("startcommontime: "+node.getStartTime());
        System.out.println("endcommontime: "+node.getEndTime());
        String participants = new String("");
        for(Integer np: node.getParticipants()){
            String block = " "+np;
            participants = participants + block;
        }
        System.out.println("participant: "+participants);
    }
}

public static void main(String[] args){
    System.out.println("Hello Java");
    for(Node d: Tree.location){
        System.out.println("Location is " + d.getLocation());
    }
    Tree.readCSV();
    for(Node d: Tree.location){
        Tree.atLeasttwo(d.getChildren());
    }
}

public ArrayList<Node> getLocation() {
    return location;
}

public void setLocation(ArrayList<Node> location) {
    this.location = location;
}
}

```

Code of Node

```
import java.util.ArrayList;
public class Node {
    private int id;
    private String startTime;
    private String endTime;
    private String location;
    private ArrayList<Node> children;
    private ArrayList<Integer> participants;
    //Construction Method
    public Node(int id, String startTime, String endTime,
String location, ArrayList<Node> children,
ArrayList<Integer> participants){
        this.setId(id);
        this.setStartTime(startTime);
        this.setEndTime(endTime);
        this.setLocation(location);
        this.setChildren(children);
        this.setParticipants(participants);
    }
    public String getLocation() {
        return location;
    }
    public void setLocation(String location) {
        this.location = location;
    }
    public ArrayList<Node> getChildren() {
        return children;
    }
    public void setChildren(ArrayList<Node> children) {
        this.children = children;
    }
    public String getEndTime() {
        return endTime;
    }
    public void setEndTime(String endTime) {
        this.endTime = endTime;
    }
    public String getStartTime() {
        return startTime;
    }
    public void setStartTime(String startTime) {
        this.startTime = startTime;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public ArrayList<Integer> getParticipants() {
        return participants;
    }
    public void setParticipants(ArrayList<Integer> participants) {
        this.participants = participants;
    }
}
```

Test Script

```
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
public class TestMeeting {
    private int id;
    private String startTime;
    private String endTime;
    private String location;
    private ArrayList<Node> children;
    private ArrayList<Integer> participants;
    public static int correct;
    public static int correct1 = 0;
    public static void readCSV() {
        String csvFileMeeting = "Meeting.csv";
        String csvFileSingle = "singlePersonEvent.csv";
        BufferedReader br = null;
        String line = "";
        String cvsSplitBy = ",";
        ArrayList<String> ls = new ArrayList<String>();
        Node locationNode = null;
        ArrayList<Node> al = new ArrayList<Node>();
        try {
            br = new BufferedReader(new FileReader(csvFileMeeting));
            line = br.readLine();
            while ((line = br.readLine()) != null) {
                String[] entry = line.split(cvsSplitBy);
                String[] participants = entry[3].split(" ");
                ArrayList<String> p = new ArrayList<String>();
                for(String s : participants){
                    s = s.replace(" ", "");
                    s = s.replace("\\", "");
                    if(s.trim().length() > 0){
                        int i = Integer.parseInt(s);
                        p.add(s);
                        BufferedReader newBr = null;
                        String newLine = "";
                        newBr = new BufferedReader(new FileReader(csvFileSingle));
                        newLine = newBr.readLine();
                        for(int ii = 0; ii < p.size(); ii++){
                            while ((newLine = newBr.readLine()) != null){
                                String[] newEntry = newLine.split(cvsSplitBy);
                                int innerInt = Integer.parseInt(newEntry[0]);
                                if(innerInt == i &&
                                    entry[2].compareTo(newEntry[4]) == 0
                                    && overlap(entry[0], entry[1], newEntry[1],
                                        newEntry[2])){
                                    System.out.println("haha " + newEntry[0] + "
                                        " + newEntry[1] + " " + newEntry[2] + " "
                                        + newEntry[3] + " " + newEntry[4]);
                                    p.remove(String.valueOf(i));
                                }
                            }
                        }
                    }
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```

        break;
    }
}
}
}
    if(p.size() > 0){
        System.out.println("meeting not right: " +
            entry[0] + " " + entry[1] +
            " " + entry[2] + " "+entry[3]);
        correct++;
    }
}

} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (br != null) {
        try {
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

}

public static void sop(Object o){
    System.out.println(o);
}

public static boolean overlap(String s1, String s2, String s11, String s22){
    if(s1.compareTo(s11) >= 0 &&
        s2.compareTo(s22) <= 0){
        return true;
    }
    else if(s1.compareTo(s11) <= 0 &&
        s2.compareTo(s22) >= 0){
        return true;
    }
    else if(s1.compareTo(s11) < 0 &&
        s2.compareTo(s11) > 0
        && s22.compareTo(s2) > 0){
        return true;
    }
    else if(s11.compareTo(s1) < 0 &&
        s22.compareTo(s1) > 0
        && s2.compareTo(s22) > 0){
        return true;
    }
    return false;
}

public static void readCSV1() {
    String csvFileMeeting = "Meeting.csv";
    String csvFileSingle = "singlePersonEvent1.csv";
    String cvsSplitBy = ",";
    ArrayList<String> ls = new ArrayList<String>();

```

```

Node locationNode = null;
ArrayList<Node> al = new ArrayList<Node>();
BufferedReader newBr = null;
String newLine = "";
try {
    newBr = new BufferedReader(new FileReader(csvFileSingle));
    newLine = newBr.readLine();
    while ((newLine = newBr.readLine()) != null){
        String[] newEntry = newLine.split(csvSplitBy);
        int innerInt = Integer.parseInt(newEntry[0]);
        BufferedReader br = null;
        String line = "";
        br = new BufferedReader(new FileReader(csvFileMeeting));
        line = br.readLine();
        while ((line = br.readLine()) != null) {
            String[] entry = line.split(csvSplitBy);
            String[] participants = entry[3].split(" ");
            ArrayList<String> p = new ArrayList<String>();
            for(String s : participants){
                s = s.replace(" ", "");
                s = s.replace("\\\"", "");
                if(s.trim().length() > 0){
                    int i = Integer.parseInt(s);
                    p.add(s);
                }
            }
            if(p.contains(newEntry[0]) &&
                entry[2].compareTo(newEntry[3]) == 0
                && overlap(entry[0], entry[1], newEntry[1],
                    newEntry[2])){
                System.out.println("haha " + newEntry[0] + " "
                    + newEntry[1] + " " + newEntry[2] + " "
                    + newEntry[3] + " " + newEntry[4]);
                correct1++;
                System.out.println(correct1);
                break;
            }
        }
    }
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (newBr != null) {
        try {
            newBr.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
}

```

```

public static void calculate() throws ParseException{
    String csvFileMeeting = "DeskMeeting.csv";
    BufferedReader br = null;
    String line = "";
    String cvsSplitBy = ",";

```

```

    long sumSeconds = 0;
    try {
        br = new BufferedReader(new FileReader(csvFileMeeting));
        line = br.readLine();
        while ((line = br.readLine()) != null) {
            String[] entry = line.split(csvSplitBy);
            System.out.println(entry[0]);
            System.out.println(entry[1]);
            DateFormat sdf = new SimpleDateFormat("hh:mm:ss");
            Date start = sdf.parse(entry[0]);
            Date end = sdf.parse(entry[1]);
            sumSeconds = sumSeconds - start.getTime() + end.getTime();
        }
        System.out.println("sum seconds: " + sumSeconds);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (br != null) {
            try {
                br.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

public static void main(String[] args) throws ParseException{
    calculate();
}
}

```


Module of Activity

```
((cl-text /Users/guxueying/Dropbox/homework/ontology/finalprojecte/activity.clif
(cl-imports C:\Users\gu\Dropbox\homework\ontology\finalprojecte\space.clif)
(cl-imports C:\Users\gu\Dropbox\homework\ontology\finalprojecte\temporalentity.clif)
(cl-imports C:\Users\gu\Dropbox\homework\ontology\finalprojecte\person.clif)
(cl-comment 'A1 Activity has three subclasses: Break, Meeting and Work')
(forall (x)
  (iff
    (Activity x)
    (or
      (Break x)
      (Meeting x)
      (Work x)
      (MultiPersonEvent x)
      (SinglePersonEvent x)
    )
  )
)
```

```
(cl-comment 'A2 Break is disjoint with Meeting and Work')
(forall (x)
  (if
    (Break x)
    (not
      (or
        (Meeting x)
        (Work x)
      )
    )
  )
)
```

```
(cl-comment 'A3 Meeting is disjoint with Break and Work')
(forall (x)
  (if
    (Meeting x)
    (not
      (or
        (Break x)
        (Work x)
      )
    )
  )
)
```

```
(cl-comment 'A4 Work is disjoint with Break and Meeting')
(forall (x)
  (if
    (Work x)
    (not
      (or
        (Break x)
        (Meeting x)
      )
    )
  )
)
```

```

(cl-comment 'A5 Break has three subclasses: OffsiteBreak, WaterBreak and WashroomBreak')
(forall (x)
  (iff
    (Break x)
    (or
      (OffsiteBreak x)
      (WaterBreak x)
      (WashroomBreak x)
    )
  )
)

```

```

(cl-comment 'A6 OffsiteBreak is disjoint with WaterBreak and WashroomBreak')
(forall (x)
  (if
    (OffsiteBreak x)
    (not
      (or
        (WaterBreak x)
        (WashroomBreak x)
      )
    )
  )
)

```

```

(cl-comment 'A7 WaterBreak is disjoint with OffsiteBreak and WashroomBreak')
(forall (x)
  (if
    (WaterBreak x)
    (not
      (or
        (OffsiteBreak x)
        (WashroomBreak x)
      )
    )
  )
)

```

```

(cl-comment 'A8 WashroomBreak is disjoint with OffsiteBreak and WaterBreak')
(forall (x)
  (if
    (WashroomBreak x)
    (not
      (or
        (OffsiteBreak x)
        (WaterBreak x)
      )
    )
  )
)

```

```

(cl-comment 'A9 Work has three subclasses: DeskWork, Visiting and TechWork')
(forall (x)
  (iff
    (Work x)
    (or
      (DeskWork x)

```

```

        (Visiting x)
        (TechWork x)
    )
)

(cl-comment 'A10 DeskWork is disjoint with Visiting and TechWork')
(forall (x)
  (if
    (DeskWork x)
    (not
      (or
        (Visiting x)
        (TechWork x)
      )
    )
  )
)

(cl-comment 'A11 Visiting is disjoint with DeskWork and TechWork')
(forall (x)
  (if
    (Visiting x)
    (not
      (or
        (DeskWork x)
        (TechWork x)
      )
    )
  )
)

(cl-comment 'A12 TechWork is disjoint with DeskWork and Visiting')
(forall (x)
  (if
    (TechWork x)
    (not
      (or
        (DeskWork x)
        (Visiting x)
      )
    )
  )
)

(cl-comment 'A13 Meeting has two subclasses: DeskMeeting and TeamMeeting')
(forall (x)
  (iff
    (Meeting x)
    (or
      (DeskMeeting x)
      (TeamMeeting x)
    )
  )
)

(cl-comment 'A14 DeskMeeting is disjoint with TeamMeeting')
(forall (x)
  (if

```

```

        (DeskMeeting x)
      (not
        (TeamMeeting x)
      )
    )
  )

(cl-comment 'A15')
(forall (m l)
  (if
    (and
      (Meeting m)
      (MeetingRoom l)
      (takeplacein m l)
    )
    (TeamMeeting m)
  )
)

(cl-comment 'A16')
(forall (m l)
  (if
    (and
      (Meeting m)
      (or
        (Cubicle l)
        (Office l)
      )
      (takeplacein m l)
    )
    (DeskMeeting m)
  )
)

(cl-comment 'A17')
(forall (x y z)
  (if
    (and
      (Activity z)
      (Space y)
      (Space x)
      (takeplacein z x)
      (takeplacein z y)
    )
    (= x y)
  )
)

(cl-comment 'A18')
(forall (z)
  (if
    (Activity z)
    (exists (x)
      (and
        (Space x)

```

```

                                (takeplacein z x)
                            )
                        )
                    )
                )
            )
        (cl-comment 'A19')
        (forall (x y)
            (if
                (takeplacein x y)
                (and
                    (Activity x)
                    (Space y)
                )
            )
        )

        (cl-comment 'A20')
        (forall (x y z)
            (if
                (and
                    (Activity z)
                    (ProperInterval y)
                    (ProperInterval x)
                    (last z x)
                    (last z y)
                )
                (= x y)
            )
        )

        (cl-comment 'A21')
        (forall (z)
            (if
                (Activity z)
                (exists (x)
                    (and
                        (ProperInterval x)
                        (last z x)
                    )
                )
            )
        )

        (cl-comment 'A22')
        (forall (x y)
            (if
                (last x y)
                (and
                    (Activity x)
                    (ProperInterval y)
                )
            )
        )

        (cl-comment 'A23')
        (forall (z)

```

```

    (if
      (Activity z)
      (exists (x)
        (and
          (Person x)
          (participate x z)
        )
      )
    )
  )
)

(cl-comment 'A24')
(forall (x y)
  (if
    (participate x y)
    (and
      (Person x)
      (Activity y)
    )
  )
)

(cl-comment 'A25')
(forall (m l)
  (if
    (and
      (MultiPersonEvent m)
      (or
        (Cubicle l)
        (Office l)
        (MeetingRoom l)
      )
    )
    (takeplacein m l)
  )
  (Meeting m)
)

(cl-comment 'A26')
(forall (m l)
  (if
    (and
      (MultiPersonEvent m)
      (MeetingRoom l)
      (takeplacein m l)
    )
    (TeamMeeting m)
  )
)

(cl-comment 'A27')
(forall (m l)
  (if
    (and
      (MultiPersonEvent m)
      (or
        (Cubicle l)
        (Office l)
      )
    )
  )
)

```

```
                (takeplacein m l)
            )
        (DeskMeeting m)
    )
)
```

Module of Space

```
(cl-text C:\Users\gu\Dropbox\homework\ontology\finalprojecte\space.clif
(cl-comment 'Space has two subclasses: Hallway and Room')
(cl-comment 'A71')
(forall (x)
  (iff
    (Space x)
    (or
      (Hallway x)
      (Room x)
    )
  )
)

(cl-comment 'A72')
(forall (x)
  (if
    (Hallway x)
    (not
      (Room x)
    )
  )
)

(cl-comment 'A73')
(cl-comment 'Room has three subclasses: BreakSpace, UtilitySpace and WorkSpace')
(forall (x)
  (iff
    (Room x)
    (or
      (BreakSpace x)
      (UtilitySpace x)
      (WorkSpace x)
    )
  )
)

(cl-comment 'A74')
(cl-comment 'BreakSpace is disjoint with UtilitySpace and WorkSpace')
(forall (x)
  (if
    (BreakSpace x)
    (not
      (or
        (UtilitySpace x)
        (WorkSpace x)
      )
    )
  )
)

(cl-comment 'A75')
(cl-comment 'UtilitySpace is disjoint with BreakSpace and WorkSpace')
(forall (x)
  (if
    (UtilitySpace x)
    (not
      (or
        (BreakSpace x)
        (WorkSpace x)
      )
    )
  )
)
```



```

    )
  )
)

(cl-comment 'A76')
(cl-comment 'WorkSpace is disjoint with BreakSpace and UtilitySpace')
(forall (x)
  (if
    (WorkSpace x)
    (not
      (or
        (BreakSpace x)
        (UtilitySpace x)
      )
    )
  )
)

(cl-comment 'A77')
(cl-comment 'BreakSpace has two subclasses: DinningRoom and Washroom')
(forall (x)
  (iff
    (BreakSpace x)
    (or
      (DinningRoom x)
      (Washroom x)
    )
  )
)

(cl-comment 'A78')
(cl-comment 'DinningRoom is disjoint with Washroom')
(forall (x)
  (if
    (DinningRoom x)
    (not
      (Washroom x)
    )
  )
)

(cl-comment 'A79')
(cl-comment 'Washroom has two subclasses: ManWashroom and WomanWashroom')
(forall (x)
  (iff
    (Washroom x)
    (or
      (ManWashroom x)
      (WomanWashroom x)
    )
  )
)

(cl-comment 'A80')
(cl-comment 'ManWashroom is disjoint with WomanWashroom')
(forall (x)
  (if
    (ManWashroom x)
    (not

```

```

        (WomanWashroom x)
    )
)

(cl-comment 'A81')
(cl-comment 'UtilitySpace has two subclasses: Photocopier and TechRoom')
(forall (x)
  (iff
    (UtilitySpace x)
    (or
      (Photocopier x)
      (TechRoom x)
    )
  )
)

(cl-comment 'A82')
(cl-comment 'TechRoom is disjoint with Photocopier')
(forall (x)
  (if
    (TechRoom x)
    (not
      (Photocopier x)
    )
  )
)

(cl-comment 'A83')
(cl-comment 'WorkSpace has five subclasses: Cubicle, CubicleSpace, MeetingRoom, Office
and VisitingRoom')
(forall (x)
  (iff
    (WorkSpace x)
    (or
      (Cubicle x)
      (CubicleSpace x)
      (MeetingRoom x)
      (Office x)
      (VisitingRoom x)
    )
  )
)

(cl-comment 'A84')
(cl-comment 'Cubicle is disjoint with CubicleSpace, MeetingRoom, Office and
VisitingRoom')
(forall (x)
  (if
    (Cubicle x)
    (not
      (or
        (CubicleSpace x)
        (MeetingRoom x)
        (Office x)
        (VisitingRoom x)
      )
    )
  )
)

```

```

(cl-comment 'A85')
(cl-comment 'CubicleSpace is disjoint with Cubicle, MeetingRoom, Office and
VisitingRoom')
(forall (x)
  (if
    (CubicleSpace x)
    (not
      (or
        (Cubicle x)
        (MeetingRoom x)
        (Office x)
        (VisitingRoom x)
      )
    )
  )
)

(cl-comment 'A86')
(cl-comment 'MeetingRoom is disjoint with Cubicle, CubicleSpace, Office and
VisitingRoom')
(forall (x)
  (if
    (MeetingRoom x)
    (not
      (or
        (Cubicle x)
        (CubicleSpace x)
        (Office x)
        (VisitingRoom x)
      )
    )
  )
)

(cl-comment 'A87')
(cl-comment 'Office is disjoint with Cubicle, CubicleSpace, MeetingRoom and
VisitingRoom')
(forall (x)
  (if
    (Office x)
    (not
      (or
        (Cubicle x)
        (CubicleSpace x)
        (MeetingRoom x)
        (VisitingRoom x)
      )
    )
  )
)

(cl-comment 'A88')
(cl-comment 'VisitingRoom is disjoint with Cubicle, CubicleSpace, MeetingRoom and
Office')
(forall (x)
  (if
    (VisitingRoom x)
    (not
      (or

```

```

(Cubicle x)
(CubicleSpace x)
(MeetingRoom x)
(Office x)
    )
  )
)

(cl-comment 'A89')
(forall (x y)
  (if
    (adjacent x y)
    (and
      (Space x)
      (Space y)
    )
  )
)

(cl-comment 'A90')
(forall (x y)
  (if
    (adjacent x y)
    (adjacent y x)
  )
)

(cl-comment 'A91')
(forall (x y)
  (if
    (adjacent x y)
    (not
      (= x y)
    )
  )
)

(cl-comment 'A92')
(forall (a b c d)
  (if
    (and
      (adjacent a b)
      (hasFloor a c)
      (hasFloor b d)
    )
    (= c d)
  )
)

(cl-comment 'A93')
(forall (a b c)
  (if
    (and
      (Space a)
      (hasFloor a c)
      (hasFloor a b)
    )
    (= c b) )))

```

Module of Person

```
(cl-text C:\Users\gu\Dropbox\homework\ontology\finalprojecte\person.clif
(cl-comment 'Person has three subclasses: Employee, Intern and Visitor')
(cl-comment 'A94')
(forall (x)
  (iff
    (Person x)
    (or
      (Employee x)
      (Intern x)
      (Visitor x)
    )
  )
)

(cl-comment 'A95')
(cl-comment 'Employee is disjoint with Intern and Visitor')
(forall (x)
  (if
    (Employee x)
    (not
      (or
        (Intern x)
        (Visitor x)
      )
    )
  )
)

(cl-comment 'A96')
(cl-comment 'Intern is disjoint with Employee and Visitor')
(forall (x)
  (if
    (Intern x)
    (not
      (or
        (Employee x)
        (Visitor x)
      )
    )
  )
)

(cl-comment 'A97')
(cl-comment 'Visitor is disjoint with Intern and Employee')
(forall (x)
  (if
    (Visitor x)
    (not
      (or
        (Intern x)
        (Employee x)
      )
    )
  )
)
)
```

Module of AllRelation

```
(cl-text C:\Users\gu\Dropbox\homework\ontology\finalprojecte\allRelation7.clif
(cl-imports C:\Users\gu\Dropbox\homework\ontology\finalprojecte\activity.clif)
(cl-imports C:\Users\gu\Dropbox\homework\ontology\finalprojecte\equipment.clif)

(cl-comment 'A56')
(forall (x)
  (if
    (Equipment x)
    (not
      (or
        (Person x)
        (Space x)
        (TemporalEntity x)
        (Activity x)
        (Department x)
        (Integer x)
      )
    )
  )
)

(cl-comment 'A57')
(cl-comment 'Person is disjoint with Equipment, Space, TemporalEntity and Activity')
(forall (x)
  (if
    (Person x)
    (not
      (or
        (Equipment x)
        (Space x)
        (TemporalEntity x)
        (Activity x)
        (Department x)
        (Integer x)
      )
    )
  )
)

(cl-comment 'A58')
(cl-comment 'Space is disjoint with Equipment, Person, TemporalEntity and Activity')
(forall (x)
  (if
    (Space x)
    (not
      (or
        (Equipment x)
        (Person x)
        (TemporalEntity x)
        (Activity x)
        (Department x)
        (Integer x)
      )
    )
  )
)
```

```

(cl-comment 'A59')
(cl-comment 'TemporalEntity is disjoint with Equipment, Person, Space and Activity')
(forall (x)
  (if
    (TemporalEntity x)
    (not
      (or
        (Equipment x)
        (Person x)
        (Space x)
        (Activity x)
        (Department x)
        (Integer x)
      )
    )
  )
)

```

```

(cl-comment 'A60')
(cl-comment 'Activity is disjoint with Equipment, Person, Space and TemporalEntity')
(forall (x)
  (if
    (Activity x)
    (not
      (or
        (Equipment x)
        (Person x)
        (Space x)
        (TemporalEntity x)
        (Department x)
        (Integer x)
      )
    )
  )
)

```

```

(cl-comment 'A61')
(forall (x)
  (if
    (Department x)
    (not
      (or
        (Equipment x)
        (Person x)
        (Space x)
        (TemporalEntity x)
        (Activity x)
        (Integer x)
      )
    )
  )
)

```

```

(cl-comment 'A62')
(forall (x)
  (if
    (Integer x)
    (not
      (or
        (Equipment x)

```

```
(Person x)
(Space x)
(TemporalEntity x)
(Activity x)
(Department x)
)
)
)
(cl-comment 'A63')
(forall (p l i)
  (if
    (PersonRoomInstant p l i)
    (and
      (Person p)
      (Space l)
      (Instant i)
    )
  )
)
(cl-comment 'A64')
(forall (p l i)
  (if
    (PersonRoomInstant p l i)
    (exists (s t)
      (and
        (SinglePersonEvent s)
        (last s t)
        (participate p s)
        (takeplacein s l)
        (inside i t)
      )
    )
  )
)
(cl-comment 'A65')
(forall (s p l t)
  (if
    (and
      (SinglePersonEvent s)
      (last s t)
      (participate p s)
      (takeplacein s l)
    )
    (exists (i)
      (and
        (inside i t)
        (PersonRoomInstant p l i)
      )
    )
  )
)
(cl-comment 'A66')
(forall (m l t)
  (if
    (and
```



```

    )
  )
)

(cl-comment 'A70')
(cl-comment 'participate(domain: person, range: activity)')
(forall (x y)
  (if
    (participate x y)
    (and
      (Person x)
      (Activity y)
    )
  )
)

)

)

(cl-comment 'A102')
(forall (x y)
  (if
    (hasdepartment x y)
    (and
      (or
        (Intern x)
        (Employee x)
      )
      (Department y)
    )
  )
)

)

(cl-comment 'A103')
(forall (x y z)
  (if
    (and
      (hasdepartment x y)
      (hasdepartment x z)
    )
    (= y z)
  )
)

)

(cl-comment 'A104')
(forall (x y z a b)
  (if
    (and
      (TeamMeeting z)
      (participate x z)
      (participate y z)
      (hasdepartment x a)
      (hasdepartment y b)
    )
    (= a b)
  )
)

)

```

Module of TemporalEntity

```
(cl-text C:\Users\gu\Dropbox\homework\ontology\finalprojecte\temporality.clif
(cl-comment 'A28 TemporalEntity has two subclasses: Instant and Interval')
(forall (x)
  (iff
    (TemporalEntity x)
    (or
      (Instant x)
      (Interval x)
    )
  )
)

(cl-comment 'A29 Interval is disjoint with Instant')
(forall (x)
  (if
    (Instant x)
    (not
      (Interval x)
    )
  )
)

(cl-comment 'A30 ProperInterval is a subclass of Interval')
(forall (x)
  (if
    (ProperInterval x)
    (Interval x)
  )
)

(cl-comment 'A31')
(forall (t1 t2)
  (if
    (before t1 t2)
    (not (= t1 t2)))
)

(cl-comment 'A32')
(forall (t1 t2)
  (if
    (before t1 t2)
    (not (before t2 t1)))
)

(cl-comment 'A33')
(forall (t1 t2)
  (if
    (before t1 t2)
    (and (Instant t1)
          (Instant t2)
    )
  )
)

(cl-comment 'A34')
(forall (t1 t2 t3)
  (if
    (and
      (before t1 t2)
      (before t2 t3))
    (before t1 t3))
)
```

```

(cl-comment 'A35')
(forall (t)
  (if (Instant t)
      (TemporalEntity t)))

(cl-comment 'A36')
(forall (t)
  (if (Interval t)
      (TemporalEntity t)))

(cl-comment 'A37')
(forall (t)
  (if (TemporalEntity t)
      (or (Interval t)
          (Instant t))))

(cl-comment 'A38')
(forall (t x)
  (if (begins t x)
      (and (Instant t)
            (TemporalEntity x))))

(cl-comment 'A39')
(forall (t x)
  (if (ends t x)
      (and (Instant t)
            (TemporalEntity x))))

(cl-comment 'A40')
(forall (t)
  (iff (Instant t)
        (begins t t)))

(cl-comment 'A41')
(forall (t)
  (iff (Instant t)
        (ends t t)))

(cl-comment 'A42')
(forall (x t1 t2)
  (if (and (TemporalEntity x)
            (begins t1 x)
            (begins t2 x))
      (= t1 t2)))

(cl-comment 'A43')
(forall (x t1 t2)
  (if (and (TemporalEntity x)
            (ends t1 x)
            (ends t2 x))
      (= t1 t2)))

(cl-comment 'A44')
(forall (x)
  (iff (ProperInterval x)
      (and (Interval x)
            (forall (t1 t2)
              (if (and (begins t1 x)
                        (ends t2 x))
                  (= t1 t2)))))

```

```

(not (= t1 t2))))))

(cl-comment 'A45')
(forall (a b ta1 ta2 tb1 tb2)
  (if
    (and
      (ProperInterval a)
      (ProperInterval b)
      (begins ta1 a)
      (ends ta2 a)
      (begins tb1 b)
      (ends tb2 b)
      (not (= a b))
      (or
        (before ta1 tb1)
        (= ta1 tb1)
      )
      (or
        (before tb2 ta2)
        (= tb2 ta2)
      )
    )
    (contain a b)
  )
)

(cl-comment 'A46')
(forall (a b ta1 ta2 tb1 tb2)
  (if
    (contain a b)
    (exists (ta1 ta2 tb1 tb2)
      (and
        (ProperInterval a)
        (ProperInterval b)
        (not (= a b))
        (begins ta1 a)
        (ends ta2 a)
        (begins tb1 b)
        (ends tb2 b)
        (or
          (before ta1 tb1)
          (= ta1 tb1)
        )
        (or
          (before tb2 ta2)
          (= tb2 ta2)
        )
      )
    )
  )
)

(cl-comment 'A47')
(forall (x t1 t2)
  (if
    (and
      (ProperInterval x)
      (begins t1 x)
      (ends t2 x)
    )
    (not (before t2 t1)))
  )
)

```

```

(cl-comment 'A48')
(forall (x t1 t2)
  (if (and (ProperInterval x)
            (begins t1 x)
            (ends t2 x))
      (before t1 t2)))

(cl-comment 'A49')
(forall (x y)
  (if
    (overlap x y)
    (overlap y x)
  )
)

(cl-comment 'A50')
(forall (x y)
  (if
    (overlap x y)
    (and
      (ProperInterval x)
      (ProperInterval y)
    )
  )
)

(cl-comment 'A51')
(forall (x y xa xb ya yb)
  (if
    (and
      (or
        (and
          (ProperInterval x)
          (ProperInterval y)
          (not (= x y))
          (begins xa x)
          (ends xb x)
          (begins ya y)
          (ends yb y)
          (before xa ya)
          (before ya xb)
          (before xb yb)
        )
        (and
          (ProperInterval x)
          (ProperInterval y)
          (not (= x y))
          (begins xa x)
          (ends xb x)
          (begins ya y)
          (ends yb y)
          (= xa ya)
          (= xb yb)
        )
      )
    )
  )
)

```


)


```

)
(cl-comment 'A55')
(forall (i t)
  (if
    (inside i t)
    (and
      (Instant i)
      (ProperInterval t)
    )
  )
)
)

```

Module of Equipment

```
(cl-text C:\Users\gu\Dropbox\homework\ontology\finalprojecte\equipment.clif
(cl-comment 'Equipment has three subclasses: Projector, Printer and Computer')
(cl-comment 'A98')
(forall (x)
  (iff
    (Equipment x)
    (or
      (Projector x)
      (Printer x)
      (Computer x)
    )
  )
)

(cl-comment 'A99')
(cl-comment 'Projector is disjoint with Printer and Computer')
(forall (x)
  (if
    (Projector x)
    (not
      (or
        (Printer x)
        (Computer x)
      )
    )
  )
)

(cl-comment 'A100')
(cl-comment 'Printer is disjoint with Projector and Computer')
(forall (x)
  (if
    (Printer x)
    (not
      (or
        (Projector x)
        (Computer x)
      )
    )
  )
)

(cl-comment 'A101')
(cl-comment 'Computer is disjoint with Printer and Projector')
(forall (x)
  (if
    (Computer x)
    (not
      (or
        (Printer x)
        (Projector x)
      )
    )
  )
)
)))
```

Consistency Result

Paradox, version 3.0, 2008-07-29.

+++ PROBLEM:

C:\Users\gu\Dropbox\homework\ontology\finalprojecte\consistency\conversions\allRelations7
_nontrivial.all.tptp

Reading

'C:\Users\gu\Dropbox\homework\ontology\finalprojecte\consistency\conversions\allRelations
7_nontrivial.all.tptp' ... OK

+++ SOLVING:

C:\Users\gu\Dropbox\homework\ontology\finalprojecte\consistency\conversions\allRelations7
_nontrivial.all.tptp

domain size 1

domain size 2

domain size 3

domain size 4

domain size 5

domain size 6

domain size 7

domain size 8

domain size 9

domain size 10

+++ BEGIN MODEL

SZS output start FiniteModel for

C:\Users\gu\Dropbox\homework\ontology\finalprojecte\consistency\conversions\allRelations7
_nontrivial.all.tptp

% domain size is 10

fof(domain, fi_domain,

(![X] : (X = "1" | X = "2" | X = "3" | X = "4" | X = "5" | X = "6" | X = "7" | X = "8"
| X = "9" | X = "10"))

).

Activity {5}

adjacent {}

before {(4, 1)}

begins{(1, 1)}

Break {}

break {}

breakspace {}

computer {}

contain {}

cubicle {}

cubiclespace {}

department {8}

deskmeeting {}

deskwork {}
dinningroom {}
employee {}
ends {(1, 1), (1, 10)}
hallway {}
equipment {6}
hasequipment {(1, 5), (3, 6)}
hasfloor ()
inside {(1, 10), (4, 10)}
instant {4}
integer {9}
intern {2}
interval {10}
last {(5, 10)}
manwashroom {}
meeting {}
meetingroom {3}
multipersonevent {}
multiplepersonevent(2)
office {}
offsitebreak {}
overlap {}
participate {(2, 5)}
person {2, 7}
personroominstant {(2, 3, 1)}
photocopier {}
printer {}
projector {6}
Properinterval {10}
room {3}

singlepersonevent {5}

space {3}

supervise {(7, 6)}

takeplacein {}

teammeeting {}

techroom {}

techwork {}

temporalentity {4, 10}

utilityspace {}

visiting {}

visitingroom {}

visitor = {7}

washroom {}

washroombreak {}

washroombreak = {}

waterrrreak = {}

womanwashroom {}

work {}

workspace {3}

SZS output end FiniteModel for

C:\Users\gu\Dropbox\homework\ontology\finalprojecte\consistency\conversions\allRelations7
_nontrivial.all.tptp

+++ END MODEL

+++ RESULT: Satisfiable

SZS status Satisfiable for

C:\Users\gu\Dropbox\homework\ontology\finalprojecte\consistency\conversions\allRelations7
_nontrivial.all.tptp

===== C:\Reasoning\paradox3 =====

execution finished: Fri Dec 11 07:30:54 2015

total CPU time used: 85.7

The command was "C:\Reasoning\paradox3 --time 600 --verbose 2 --model --tstp

C:\Users\gu\Dropbox\homework\ontology\finalprojecte\consistency\conversions\allRelations7
_nontrivial.all.tptp"

Input read from

===== end of footer =====