Information Systems
Project Report

Modelling TripAdvisor

Onno de Gouw
Stefan Popa

07-01-2020

Table of contents

Chapter 1

1.1 Introduction

In computing science, developing a large software application is a difficult task. In order to get a better overview of what your application needs to be able to do and in order to get a more structured view of the application that you are going to make, it is possible to create an information system for it. This will help you in understanding what you are creating. However, as the applications tend to become more and more complex, the development of your information system also becomes more complex. It is hard to come up with a reliable information system that is easy to read and meets users' expectations.
Methods have been introduced in order to improve the quality of information systems and to improve the productivity of developers and in this report we will use those methods.
We are going to put those methods into practice and test them against an existing real-life complex software application. We will determine the strengths and the weaknesses of the theory and try to explore its boundaries by picking fragments of the real-life application and coming up with advanced and interesting models for those fragments. The application that we will look at in this report will be TripAdvisor, the website that can be found at www.tripadvisor.com.
After creating the models of the fragments of TripAdvisor, we will discuss the strengths and weaknesses in Chapter 3 - Part B of this report and finally we will discuss some possible extensions and improvements to the theory in Chapter 4 – Part C of this report.

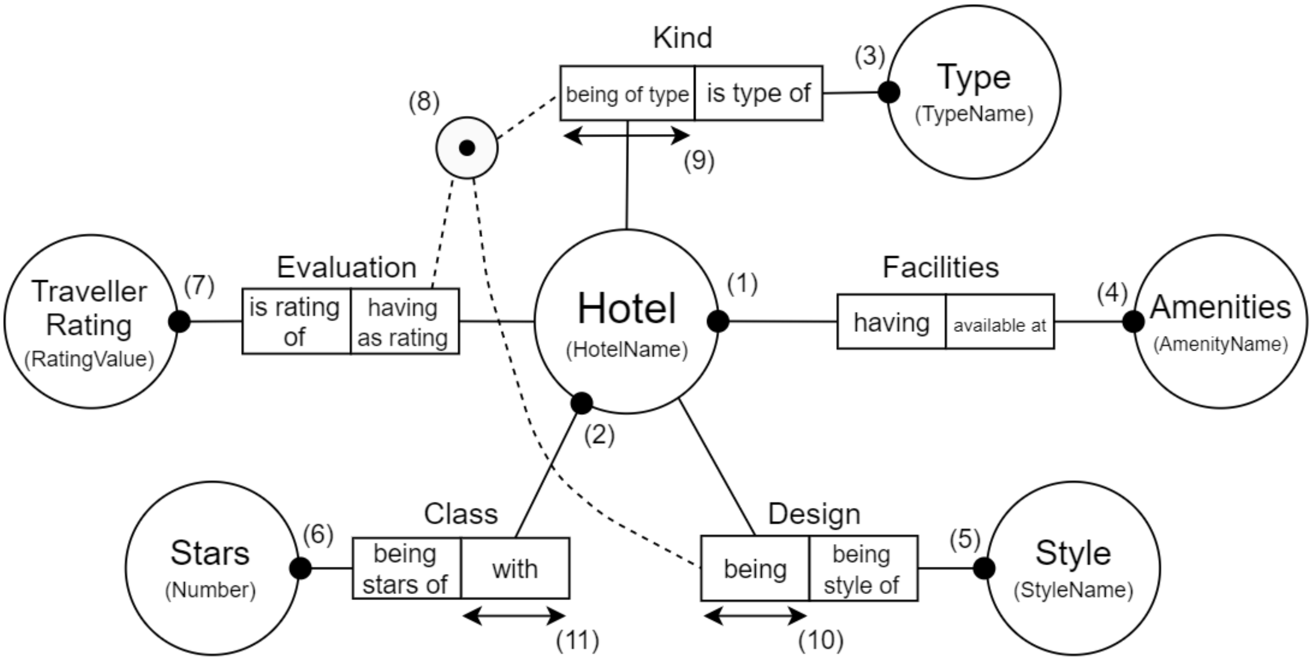
Chapter 2 – Part A

2.1 Introduction

In this chapter, we are going to show the models that we created in order to model fragments of the website TripAdvisor. We include their mathematical formalizations as well and also the populations that we created for them. The fragments of the website that we modelled are the hotel filtering system, the flight search system, the cruise information system and a fragment of the overview of things you can do in some area. The idea is that the further you get in this chapter, the more complex the model fragments will get.

2.2 Hotel filtering system

2.2.1 Diagram



2.2.2 Formal definition

$\mathcal{E} = \{\text{Hotel}, \text{Type}, \text{Traveller Rating}, \text{Stars}, \text{Style}, \text{Amenities}\}$

$\mathcal{F} = \{\text{Kind}, \text{Evaluation}, \text{Class}, \text{Design}, \text{Facilities}\}$

$\mathcal{P} = \{\text{being of type}, \text{is type of}, \text{having as rating}, \text{is rating of}, \text{with}, \text{being stars of}, \text{being},$
$\text{being style of}, \text{having}, \text{available at}\}$

$\mathcal{L} = \{\text{HotelName}, \text{TypeName}, \text{RatingValue}, \text{Number}, \text{StyleName}, \text{AmenityName}\}$

$\mathcal{O} = \mathcal{E} \cup \mathcal{F} \cup \mathcal{L}$

$\text{Kind} = \{\text{being of type}, \text{is type of}\}$

$\text{Evaluation} = \{\text{having as rating}, \text{is rating of}\}$

$\text{Class} = \{\text{with}, \text{being stars of}\}$

$\text{Design} = \{\text{being}, \text{being style of}\}$

$\text{Amenities} = \{\text{having}, \text{available at}\}$

$\text{Base}(\text{being of type}) = \text{Base}(\text{having as rating}) = \text{Base}(\text{with}) = \text{Base}(\text{being})$
$= \text{Base}(\text{having}) = \text{Hotel}$

$\text{Base}(\text{is type of}) = \text{Type}$

$\text{Base}(\text{is rating of}) = \text{Traveller Rating}$

$\text{Base}(\text{being stars of}) = \text{Stars}$

$\text{Base}(\text{being style of}) = \text{Style}$

$\text{Base}(\text{available at}) = \text{Amenities}$

Formal definition of the constraints:
First we show type relatedness within our model:
being of type ~ having ~ being ~ with ~ having as rating

In order to specify the formal definition of the constraints of this model we added numbers to the constraints in the drawing such that we can refer to them and we take the following steps for every constraint:

1. Explain informally what the constraint is supposed to do.
2. Give the formal semantics.
3. Create an illegal population.

Constraint 1:

1. This constraint is supposed to specify the fact that each instance of the *Hotel* object type must play the role *having*. In other words this means that all hotels shown in the search of TripAdvisor should have at least 1 amenity.
2. total({having})
3. Suppose we have the following:
   Pop(Hotel) = {Hotel Bellerive, Duven Hotel}
   Pop(Amenities) = {Free Wifi, Pool}

Hotel Bellerive	Free Wifi
Hotel Bellerive	Pool

This is an illegal population, because Duven Hotel does not have any amenities.

Constraint 2:

1. This constraint is supposed to specify the fact that each instance of the *Hotel* object type must play the role *with*. In other words this means that all hotels shown in the search of TripAdvisor should have a star rating.
2. total({with})
3. Suppose we have the following:
   Pop(Hotel) = {Hotel Bellerive, Duven Hotel}
   Pop(Stars) = {3}

Hotel Bellerive	3

This is an illegal population, because Duven Hotel does not have a star rating.

Constraint 3:

1. This constraint is supposed to specify the fact that each instance of the *Type* object type must play the role *is type of*. In other words this means that all types available in the search filter of TripAdvisor should be associated with at least 1 hotel.
2. total({is type of})
3. Suppose we have the following:
   Pop(Type) = {Hotel, Resort}
   Pop(Hotel) = {Hotel Bellerive, Duven Hotel}

Hotel	Hotel Bellerive
Hotel	Duven Hotel

This is an illegal population, because Resort is not associated with any hotel.

Constraint 4:

1. This constraint is supposed to specify the fact that each instance of the *Amenities* object type must play the role *available at*. In other words this means that all amenities available in the search filter of TripAdvisor should be associated with at least 1 hotel.
2. total({available at})
3. Suppose we have the following:
   Pop(Amenities) = {Free Wifi, Pool}
   Pop(Hotel) = {Hotel Bellerive, Duven Hotel}

Free Wifi	Hotel Bellerive
Free Wifi	Duven Hotel

This is an illegal population, because Pool is not available at any hotel.

Constraint 5:

1. This constraint is supposed to specify the fact that each instance of the *Style* object type must play the role *being style of*. In other words this means that all styles available in the search filter of TripAdvisor should be associated with at least 1 hotel.
2. total({being style of})
3. Suppose we have the following:
   Pop(Style) = {Modern, Luxury}
   Pop(Hotel) = {Hotel Bellerive, Duven Hotel}

Modern	Hotel Bellerive
Modern	Duven Hotel

This is an illegal population, because Luxury is not associated with any hotel.

Constraint 6:

1. This constraint is supposed to specify the fact that each instance of the *Stars* object type must play the role *being stars of*. In other words this means that all star ratings available in the search filter of TripAdvisor should be associated with at least 1 hotel.
2. total({being stars of})
3. Suppose we have the following:
   Pop(Stars) = {3, 4}
   Pop(Hotel) = {Hotel Bellerive, Duven Hotel}

4	Hotel Bellerive
4	Duven Hotel

This is an illegal population, because the star rating 3 is not given to any hotel.

Constraint 7:

1. This constraint is supposed to specify the fact that each instance of the *Traveller Rating* object type must play the role *is rating of*. In other words this means that all traveller ratings available in the search filter of TripAdvisor should be associated with at least 1 hotel.
2. total({is rating of})
3. Suppose we have the following:
   Pop(Traveller Rating) = {Excellent, Good}
   Pop(Hotel) = {Hotel Bellerive, Duven Hotel}

Excellent	Hotel Bellerive
Excellent	Duven Hotel

   This is an illegal population, because traveller rating Good is not given to any hotel.

Constraint 8:

1. This constraint is supposed to specify the fact that each instance of the *Hotel* object type must play either one of the roles *being of type, having as rating* and *being*, two of those roles or all of them. In other words this means that all hotels shown in the search of TripAdvisor should have a type and/or a traveller rating and/or a style.
2. total({being of type, having as rating, being}). (Allowed because all are type related.)
3. Suppose we have the following:
   Pop(Hotel) = {Hotel Bellerive, Duven Hotel, The Hari}
   Pop(Type) = {Hotel}
   Pop(Style) = {Modern}
   Pop(Traveller Rating) = {Excellent}

Hotel Bellerive	Hotel
Duven Hotel	Modern
Duven Hotel	Excellent

   This is an illegal population, because The Hari does not play a role in any of the three fact types.

Constraint 9:

1. This constraint is supposed to specify the fact that each instance of the *Hotel* object type plays the role *being of type* at most once. In other words this means that each hotel shown in the search of TripAdvisor should not have more than 1 type.
2. unique({being of type})
   identifier({being of type}, Kind)
3. Suppose we have the following:
   Pop(Hotel) = {Hotel Bellerive}
   Pop(Type) = {Hotel, Resort}

Hotel Bellerive	Hotel
Hotel Bellerive	Resort

   This is an illegal population, because Hotel Bellerive has more than one type.

Constraint 10:

1. This constraint is supposed to specify the fact that each instance of the *Hotel* object type plays the role *being* at most once. In other words this means that each hotel shown in the search of TripAdvisor should not have more than 1 style.
2. unique({being})
   identifier({being}, Design)
3. Suppose we have the following:
   Pop(Hotel) = {Hotel Bellerive}
   Pop(Style) = {Modern, Luxury}

Hotel Bellerive	Modern
Hotel Bellerive	Luxury

   This is an illegal population, because Hotel Bellerive has more than one style.

Constraint 11:

1. This constraint is supposed to specify the fact that each instance of the *Hotel* object type plays the role *with* at most once. In other words this means that each hotel shown in the search of TripAdvisor should not have more than 1 star rating.
2. unique({with})
   identifier({with}, Class)
3. Suppose we have the following:
   Pop(Hotel) = {Hotel Bellerive}
   Pop(Stars) = {3, 4}

Hotel Bellerive	3
Hotel Bellerive	4

   This is an illegal population, because Hotel Bellerive has more than one star rating.

2.2.3 Population

We are allowed to use the instances of the label types as instances for the entity types for the label types that are treated as identifiers for the entity types. We can do this because there is a bijection between those two. Therefore, we will use this to shorten our definition of the population. We will use this shortened definition in all the next models as well.

$\text{Pop(Hotel)} = \{\text{Hotel Bellerive}, \text{Duven Hotel}, \text{The Hari}, \text{Hotel Firefly}\}$

$\text{Pop(Type)} = \{\text{Hotel}\}$

$\text{Pop(Traveler Rating)} = \{\text{Excellent}, \text{Good}\}$

$\text{Pop(Stars)} = \{3, 4, 5\}$

$\text{Pop(Style)} = \{\text{Modern}, \text{FamilyFriendly}, \text{Luxury}\}$

$\text{Pop(Amenities)} = \{\text{Free Wifi}, \text{Pool}, \text{Massage}\}$

$\text{Pop(Kind)} = \{t_1, t_2, t_3, t_4\}$

$t_1(\text{being of type}) = \text{Hotel Bellerive}$

$t_1(\text{is type of}) = \text{Hotel}$

$t_2(\text{being of type}) = \text{Duven Hotel}$

$t_2(\text{is type of}) = \text{Hotel}$

$t_3(\text{being of type}) = \text{The Hari}$

$t_3(\text{is type of}) = \text{Hotel}$

$t_4(\text{being of type}) = \text{Hotel Firefly}$

$t_4(\text{is type of}) = \text{Hotel}$

$\text{Pop(Evaluation)} = \{t_5, t_6, t_7, t_8\}$

$t_5(\text{is rating of}) = \text{Excellent}$

$t_5(\text{having as rating}) = \text{Hotel Bellerive}$

$t_6(\text{is rating of}) = \text{Good}$

$t_6(\text{having as rating}) = \text{Duven Hotel}$

$t_7(\text{is rating of}) = \text{Excellent}$

$t_7(\text{having as rating}) = \text{The Hari}$

$t_8(\text{is rating of}) = \text{Excellent}$

$t_8(\text{having as rating}) = \text{Hotel Firefly}$

$\text{Pop(Class)} = \{t_9, t_{10}, t_{11}, t_{12}\}$

$t_9(\text{being stars of}) = 3$

$t_9(\text{with}) = \text{Hotel Bellerive}$

$t_{10}(\text{being stars of}) = 3$

$t_{10}(\text{with}) = \text{Duven Hotel}$

$t_{11}(\text{being stars of}) = 5$

$t_{11}(\text{with}) = \text{The Hari}$

$t_{12}(\text{being stars of}) = 4$

$t_{12}(\text{with}) = \text{Hotel Firefly}$

$\text{Pop(Design)} = \{t_{13}, t_{14}, t_{15}, t_{16}\}$

$t_{13}(\text{being}) = \text{Modern}$

$t_{13}(\text{being style of}) = \text{Hotel Bellerive}$

$t_{14}(\text{being}) = \text{FamilyFriendly}$

$t_{14}(\text{being style of}) = \text{Duven Hotel}$

$t_{15}(\text{being}) = \text{Modern}$

$t_{15}(\text{being style of}) = \text{The Hari}$

$t_{16}(\text{being}) = \text{Luxury}$

$t_{16}(\text{being style of}) = \text{Hotel Firefly}$

$\text{Pop(Facilities)} = \{t_{17}, t_{18}, t_{19}, t_{20}, t_{21}, t_{22}, t_{23}, t_{24}\}$

$t_{17}(\text{having}) = \text{Hotel Bellerive}$

$t_{17}(\text{available at}) = \text{Free Wifi}$

$t_{18}(\text{having}) = \text{Duven Hotel}$

$t_{18}(\text{available at}) = \text{Free Wifi}$

$t_{19}(\text{having}) = \text{The Hari}$

$t_{19}(\text{available at}) = \text{Free Wifi}$

$t_{20}(\text{having}) = \text{Hotel Firefly}$

$t_{20}(\text{available at}) = \text{Free Wifi}$

$t_{21}(\text{having}) = \text{Hotel Bellerive}$

$t_{21}(\text{available at}) = \text{Pool}$

$t_{22}(\text{having}) = \text{The Hari}$

$t_{22}(\text{available at}) = \text{Massage}$

$t_{23}(\text{having}) = \text{Hotel Firefly}$

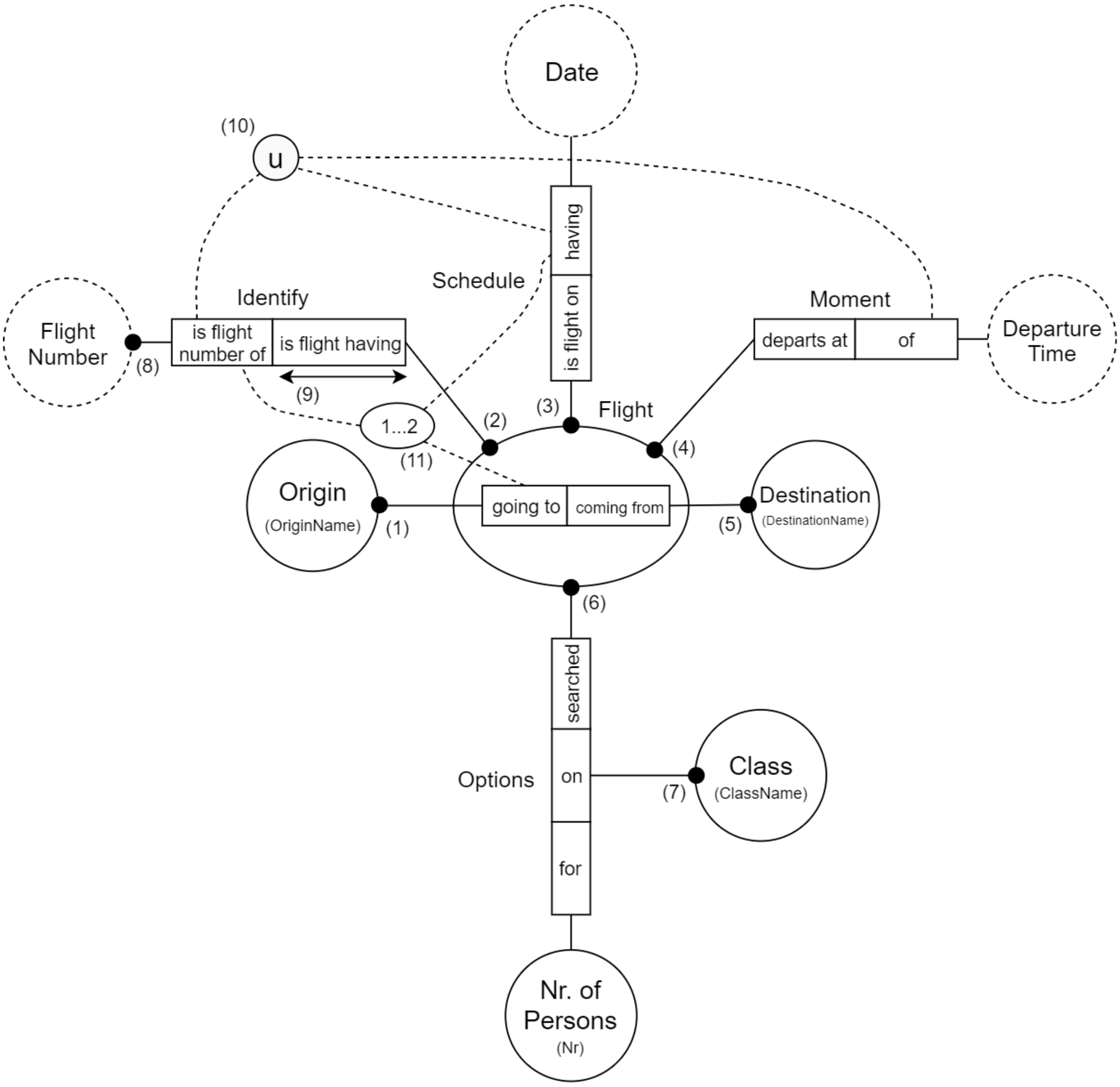
$t_{23}(\text{available at}) = \text{Massage}$

$t_{24}(\text{having}) = \text{Hotel Firefly}$

$t_{24}(\text{available at}) = \text{Pool}$

2.3 Flight search system – One way

2.3.1 Diagram



2.3.2 Formal definition

$\mathcal{E} = \{\text{Origin, Destination, Class, Nr. of Persons}\}$

$\mathcal{F} = \{\text{Identify, Schedule, Moment, Flight, Options}\}$

$\mathcal{P} = \{$is flight number of, is flight having, having, is flight on, departs at, of, going to coming from, searched, on, for$\}$

$\mathcal{L} = \{$Flight Number, Date, Departure Time, OriginName, DestinationName, ClassName, Nr$\}$

$\mathcal{O} = \mathcal{E} \cup \mathcal{F} \cup \mathcal{L}$

Identify $= \{$is flight number of, is flight having$\}$

$\text{Schedule} = \{\text{having}, \text{is flight on}\}$

$\text{Moment} = \{\text{departs at}, \text{of}\}$

$\text{Flight} = \{\text{going to}, \text{coming from}\}$

$\text{Options} = \{\text{searched}, \text{on}, \text{for}\}$

$\text{Base}(\text{is flight having}) = \text{Base}(\text{is flight on}) = \text{Base}(\text{departs at}) = \text{Base}(\text{searched}) = \text{Flight}$

$\text{Base}(\text{is flight having}) = \text{Flight Number}$

$\text{Base}(\text{having}) = \text{Date}$

$\text{Base}(\text{of}) = \text{Departure Time}$

$\text{Base}(\text{going to}) = \text{Origin}$

$\text{Base}(\text{coming from}) = \text{Destination}$

$\text{Base}(\text{on}) = \text{Class}$

$\text{Base}(\text{for}) = \text{Nr. of Persons}$

Formal definition of the constraints:
First we show type relatedness within our model:
is flight having ~ is flight on ~ departs at ~ searched

In order to specify the formal definition of the constraints of this model we added numbers to the constraints in the drawing such that we can refer to them and we take the following steps for every constraint:

1. Explain informally what the constraint is supposed to do.
2. Give the formal semantics.
3. Create an illegal population.

Constraint 1:

1. This constraint is supposed to specify the fact that each instance of the *Origin* object type must play the role *going to*. In other words this means that all departure locations available in the flight search of TripAdvisor should have at least one available destination.
2. total({going to})
3. Suppose we have the following:
   Pop(Origin) = {Amsterdam, Bucharest}
   Pop(Destination) = {New York City, Stockholm}

Amsterdam	New York City
Amsterdam	Stockholm

This is an illegal population, because Bucharest is never used as departure location.

Constraint 2:

1. This constraint is supposed to specify the fact that each instance of the *Flight* object type must play the role *is flight having*. In other words this means that all available flights in the flight search of TripAdvisor should have at least one flight number.
2. total({is flight having})
3. Suppose we have the following:
   Pop(Flight) = $\{t_1, t_2\}$
   t_1(going to) = Amsterdam t_2(going to) = Amsterdam
   t_1(coming from) = Stockholm t_2(coming from) = New York City
   Pop(Flight Number) = {DY4258}

t_1	DY4258

   This is an illegal population, because flight t_2 does not have a flight number.

Constraint 3:

1. This constraint is supposed to specify the fact that each instance of the *Flight* object type must play the role *is flight on*. In other words this means that all available flights in the flight search of TripAdvisor should have at least one departure date.
2. total({is flight on})
3. Suppose we have the following:
   Pop(Flight) = $\{t_1, t_2\}$
   t_1(going to) = Amsterdam t_2(going to) = Amsterdam
   t_1(coming from) = Stockholm t_2(coming from) = New York City
   Pop(Date) = {Mon 12/9}

t_1	Mon 12/9

   This is an illegal population, because flight t_2 does not have a departure date.

Constraint 4:

1. This constraint is supposed to specify the fact that each instance of the *Flight* object type must play the role *departs at*. In other words this means that all available flights in the flight search of TripAdvisor should have at least one departure time.
2. total({departs at})
3. Suppose we have the following:
   Pop(Flight) = $\{t_1, t_2\}$
   t_1(going to) = Amsterdam t_2(going to) = Amsterdam
   t_1(coming from) = Stockholm t_2(coming from) = New York City
   Pop(Departure Time) = {08:25}

t_1	08:25

   This is an illegal population, because flight t_2 does not have a departure time.

Constraint 5:

1. This constraint is supposed to specify the fact that each instance of the *Destination* object type must play the role *coming from*. In other words this means that all available destinations in the flight search of TripAdvisor should have at least one departure location.
2. total({coming from})
3. Suppose we have the following:
   Pop(Destination) = {New York City, Stockholm}
   Pop(Origin) = {Amsterdam, Bucharest}

New York City	Amsterdam
New York City	Bucharest

This is an illegal population, because Stockholm is never used as a destination.

Constraint 6:

1. This constraint is supposed to specify the fact that each instance of the *Flight* object type must play the role *searched*. In other words this means that all available flights in the flight search of TripAdvisor should have at least one class available and a certain number of persons available for booking.
2. total({searched})
3. Suppose we have the following:
   Pop(Flight) = $\{t_1, t_2\}$
   t_1(going to) = Amsterdam t_2(going to) = Amsterdam
   t_1(coming from) = Stockholm t_2(coming from) = New York City
   Pop(Class) = {Economy, Business Class}
   Pop(Nr. Of Persons) = {3}

t_1	Economy	3
t_2	Business Class	

This is an illegal population, because flight t_2 does not have a number of persons available for booking.

Constraint 7:

1. This constraint is supposed to specify the fact that each instance of the *Class* object type must play the role *on*. In other words this means that all possible classes in the flight search of TripAdvisor should be associated with at least one flight and a certain number of persons available for booking.
2. total({on})
3. Suppose we have the following:
   Pop(Class) = {Economy, Business Class}
   Pop(Flight) = $\{t_1, t_2\}$
   t_1(going to) = Amsterdam t_2(going to) = Amsterdam
   t_1(coming from) = Stockholm t_2(coming from) = New York City
   Pop(Nr. Of Persons) = {3, 4}

Economy	t_1	3
Economy	t_2	4

This is an illegal population, because Business Class is not used for any flight.

Constraint 8:

1. This constraint is supposed to specify the fact that each instance of the *Flight* label type must play the role *is flight number of*. In other words this means that all possible flight numbers in the flight search of TripAdvisor should be associated with at least one flight.
2. total({is flight number of})
3. Suppose we have the following:
   Pop(Flight Number) = {DY4528, KL6118}
   Pop(Flight) = $\{t_1, t_2\}$
   t_1(going to) = Amsterdam
   t_1(coming from) = Stockholm
   t_2(going to) = Amsterdam
   t_2(coming from) = New York City

DY4528	t_1
DY4528	t_2

This is an illegal population, because flight number KL6118 is never used for any flight.

Constraint 9:

1. This constraint is supposed to specify the fact that each instance of the *Flight* object type plays the role *is flight having* at most once. In other words this means that each available flight should not have more than one flight number.
2. unique({is flight having})
   identifier({is flight having}, Identify)
3. Suppose we have the following:
   Pop(Flight) = $\{t_1\}$
   t_1(going to) = Amsterdam
   t_1(coming from) = Stockholm
   Pop(Flight Number) = {DY4528, KL6118}

t_1	DY4528
t_1	KL6118

This is an illegal population, because flight t_1 has more than one flight number.

Constraint 10:

1. This constraint is supposed to specify the fact that each combination of the *Flight Number*, *Date* and *Departure Time* label types should play a role with the *Flight* object type at most once. In other words this means that each flight shown in the search of TripAdvisor should not have the same departure data twice.
2. unique({is flight number of, having, of})
   identifier({is flight number of, having, of}, ξ({is flight number of, having, of}))
3. Suppose we have the following:
   Pop(Flight) = $\{t_1, t_2\}$
   t_1(going to) = Amsterdam
   t_1(coming from) = Stockholm
   t_2(going to) = Amsterdam
   t_2(coming from) = New York City
   Pop(Flight Number) = {DY4528}
   Pop(Date) = {Mon 12/9}
   Pop(Departure Time) = {08:25}

t_1	DY4528	Mon 12/9	08:25
t_2	DY4528	Mon 12/9	08:25

This is an illegal population, because the same departure data is used twice for different flights.

Constraint 11:

1. This constraint is supposed to specify the fact that each combination of the *Origin* object type and the *Flight Number* and *Date* label types should appear at least once and at most twice. In other words this means that a flight with a certain origin cannot fly from that departure location more than twice on the same date.
2. frequency({going to, is flight number of, having}, 1, 2)
   identifier({going to, is flight number of, having},
   ξ({going to, is flight number of, having}))
   Here ξ means unnesting on Identify and Schedule.
3. Suppose we have the following:
   Pop(Origin) = {Amsterdam}
   Pop(Destination) = {Stockholm}
   Pop(Flight Number) = {DY4528}
   Pop(Date) = {Mon 12/9}

Amsterdam	Stockholm	DY4528	Mon 12/9
Amsterdam	Stockholm	DY4528	Mon 12/9
Amsterdam	Stockholm	DY4528	Mon 12/9

This is an illegal population, because the same flight with a certain origin flies from the same departure location more than twice on the same date.

2.3.3 Population

$\text{Pop(Flight Number)} = \{\text{DY4258, KL6118, RO331}\}$

$\text{Pop(Date)} = \{\text{Mon 12/9}, \text{Wed 1/8}\}$

$\text{Pop(Departure Time)} = \{\text{08:25, 16:00, 17:10}\}$

$\text{Pop(Origin)} = \{\text{Amsterdam}, \text{Bucharest}\}$

$\text{Pop(Destination)} = \{\text{New York City}, \text{Stockholm}\}$

$\text{Pop(Class)} = \{\text{Economy}, \text{Business Class}, \text{First Class}\}$

$\text{Pop(Nr. of Persons)} = \{1, 3, 4\}$

$\text{Pop(Flight)} = \{t_1, t_2, t_3\}$

$t_1(\text{going to}) = \text{Amsterdam}$

$t_1(\text{coming from}) = \text{Stockholm}$

$t_2(\text{going to}) = \text{Amsterdam}$

$t_2(\text{coming from}) = \text{New York City}$

$t_3(\text{going to}) = \text{Bucharest}$

$t_3(\text{coming from}) = \text{Stockholm}$

Pop(Identify) = $\{t_4, t_5, t_6\}$

t_4(is flight number of) = DY4258

t_4(is flight having) = t_1

t_5(is flight number of) = KL6118

t_5(is flight having) = t_2

t_6(is flight number of) = RO331

t_6(is flight having) = t_3

Pop(Schedule) = $\{t_7, t_8, t_9\}$

t_7(is flight on) = t_1

t_7(having) = Mon 12/9

t_8(is flight on) = t_2

t_8(having) = Wed 1/8

t_9(is flight on) = t_3

t_9(having) = Mon 12/9

Pop(Moment) = $\{t_{10}, t_{11}, t_{12}\}$

t_{10}(departs at) = t_1

t_{10}(of) = 16:00

t_{11}(departs at) = t_2

t_{11}(of) = 17:10

t_{12}(departs at) = t_3

t_{12}(of) = 08:25

Pop(Options) = $\{t_{13}, t_{14}, t_{15}\}$

t_{13}(searched) = t_2

t_{13}(on) = Economy

t_{13}(for) = 4

t_{14}(searched) = t_1

t_{14}(on) = Business Class

t_{14}(for) = 3

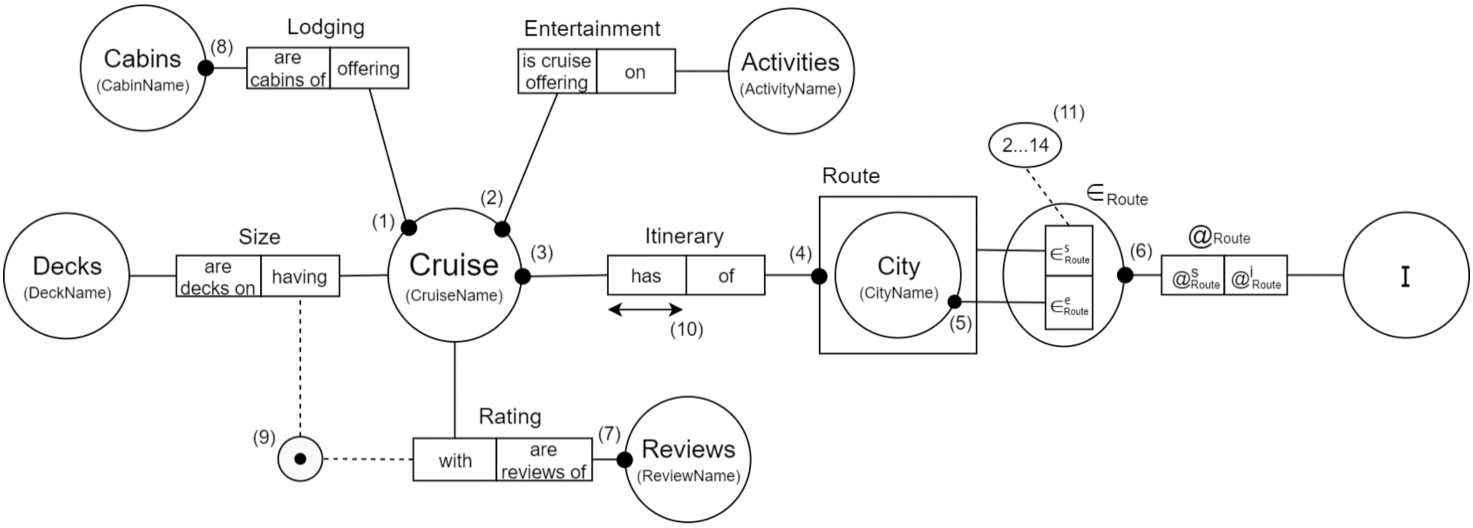
t_{15}(searched) = t_3

t_{15}(on) = First Class

t_{15}(for) = 1

2.4 Cruise information system

2.4.1 Diagram



2.4.2 Formal definition

$\mathcal{E} = \{\text{Cruise, Cabins, Activities, Route, City, Reviews, Decks}\}$

$\mathcal{F} = \{\text{Entertainment, Itinerary}, \in_{\text{Route}}, @_{\text{Route}}, \text{Rating, Size, Lodging}\}$

$\mathcal{S} = \{\text{Route}\}$

$\mathcal{P} = \{\text{is cruise offering, on, has, of}, \in^{\text{e}}_{\text{Route}}, \in^{\text{s}}_{\text{Route}}, @^{\text{s}}_{\text{Route}}, @^{\text{i}}_{\text{Route}}, \text{with, are reviews of,}$
$\text{having, are decks on, offering, are cabins of}\}$

$\mathcal{L} = \{\text{CruiseName, CabinName, ActivityName, CityName, I, ReviewName, DeckName}\}$

$\mathcal{O} = \mathcal{E} \cup \mathcal{F} \cup \mathcal{S} \cup \mathcal{L}$

$\text{Lodging} = \{\text{are cabins of, offering}\}$

$\text{Entertainment} = \{\text{is cruise offering, on}\}$

$\text{Itinerary} = \{\text{has, of}\}$

$\in_{\text{Route}} = \{\in^{\text{s}}_{\text{Route}}, \in^{\text{e}}_{\text{Route}}\}$

$@_{\text{Route}} = \{@^{\text{s}}_{\text{Route}}, @^{\text{i}}_{\text{Route}}\}$

$\text{Rating} = \{\text{with, are reviews of}\}$

$\text{Size} = \{\text{are decks on, having}\}$

$\text{Base(offering)} = \text{Base(is cruise offering)} = \text{Base(has)} = \text{Base(with)}$

$= \text{Base(having)} = \text{Cruise}$

$\text{Base(are cabins of)} = \text{Cabins}$

$\mathrm{Base(on)} = \mathrm{Activities}$

$\mathrm{Base(of)} = \mathrm{Route}$

$\mathrm{Base(are\ reviews\ of)} = \mathrm{Reviews}$

$\mathrm{Base(are\ decks\ on)} = \mathrm{Decks}$

$\mathrm{Base}(\in^{\mathrm{s}}_{\mathrm{Route}}) = \mathrm{Route}$

$\mathrm{Base}(\in^{\mathrm{e}}_{\mathrm{Route}}) = \mathrm{City}$

$\mathrm{Base}(@^{\mathrm{s}}_{\mathrm{Route}}) = \in_{\mathrm{Route}}$

$\mathrm{Base}\left(@^{\mathrm{i}}_{\mathrm{Route}}\right) = \mathrm{I}$

$\mathrm{Elt(Route)} = \mathrm{City}$

Formal definition of the constraints:
First we show type relatedness within our model:
offering ~ is cruise offering ~ has ~ with ~ having
Route ~ City

In order to specify the formal definition of the constraints of this model we added numbers to the constraints in the drawing such that we can refer to them and we take the following steps for every constraint:

1. Explain informally what the constraint is supposed to do.
2. Give the formal semantics.
3. Create an illegal population.

Constraint 1:

1. This constraint is supposed to specify the fact that each instance of the *Cruise* object type must play the role *offering*. In other words this means that all cruises shown on TripAdvisor should offer at least one type of cabin.
2. total({offering})
3. Suppose we have the following:
   Pop(Cruise) = {7 Night Cruise to the Western Mediterranean,
   4 Night Cruise to Europe}
   Pop(Cabins) = {Suite, Balcony}

4 Night Cruise to Europe	Suite
4 Night Cruise to Europe	Balcony

This is an illegal population, because the cruise 7 Night Cruise to the Western Mediterranean does not have any type of cabins.

Constraint 2:

1. This constraint is supposed to specify the fact that each instance of the *Cruise* object type must play the role *is cruise offering*. In other words this means that all cruises shown on TripAdvisor should offer at least one activity.
2. total({is cruise offering})
3. Suppose we have the following:
   Pop(Cruise) = {7 Night Cruise to the Western Mediterranean,
   4 Night Cruise to Europe}
   Pop(Activities) = {Gym, Casino}

4 Night Cruise to Europe	Gym
4 Night Cruise to Europe	Casino

This is an illegal population, because the cruise 7 Night Cruise to the Western Mediterranean does not offer any activities.

Constraint 3:

1. This constraint is supposed to specify the fact that each instance of the *Cruise* object type must play the role *has*. In other words this means that all cruises shown on TripAdvisor should have at least one route.
2. total({has})
3. Suppose we have the following:
   Pop(Cruise) = {7 Night Cruise to the Western Mediterranean,
   4 Night Cruise to Europe}
   Pop(Route) = {R_1, R_2}
   R_1 = (Southampton, IJmuiden, IJmuiden, Southampton)
   R_2 = (Tenerife, Las Palmas, San Sebastian, Funchal, Lanzarote, Puerto del Rosario, Tenerife)

4 Night Cruise to Europe	R_1
4 Night Cruise to Europe	R_2

This is an illegal population, because the cruise 7 Night Cruise to the Western Mediterranean does not have any route.

Constraint 4:

1. This constraint is supposed to specify the fact that each instance of the *Route* sequence type must play the role *of*. In other words this means that all available routes shown on TripAdvisor should be associated with at least one cruise.
2. total({of})
3. Suppose we have the following:
   Pop(Route) = {R_1, R_2}
   R_1 = (Southampton, IJmuiden, IJmuiden, Southampton)
   R_2 = (Tenerife, Las Palmas, San Sebastian, Funchal, Lanzarote, Puerto del Rosario, Tenerife)
   Pop(Cruise) = {7 Night Cruise to the Western Mediterranean,
   4 Night Cruise to Europe}

R_1	7 Night Cruise to the Western Mediterranean
R_1	4 Night Cruise to Europe

This is an illegal population, because route R_2 is not associated with any cruise.

Constraint 5:

1. This constraint is supposed to specify the fact that each instance of the *City* object type must play the role $\in^{e}_{Route}$. In other words this means that all cities available for cruises shown on TripAdvisor should be associated with at least one route.
2. total($\{\in^{\mathrm{e}}_{\mathrm{Route}}\}$)
3. Suppose we have the following:
   Pop(City) = {Southampton, IJmuiden, Tenerife}
   Pop(Route) = {R_1}
   R_1 = (Southampton, IJmuiden, IJmuiden, Southampton)

Southampton	R_1
IJmuiden	R_1

This is an illegal population, because Tenerife does not appear in any available route.

Constraint 6:

1. This constraint is supposed to specify the fact that each instance of the $\in_{Route}$ object type must play the role $@^{s}_{Route}$. In other words this means that all cities available for cruises shown on TripAdvisor should have at least one position in the route that they are part of.
2. total($\{@^{\mathrm{s}}_{\mathrm{Route}}\}$)
3. Suppose we have the following:
   Pop($\in_{\mathrm{Route}}$) = {t_1, t_2, t_3}
   $t_1(\in^{\mathrm{s}}_{\mathrm{Route}})$ = R_1
   $t_1(\in^{\mathrm{e}}_{\mathrm{Route}})$ = Southampton

   $t_2(\in^{\mathrm{s}}_{\mathrm{Route}})$ = R_1
   $t_2(\in^{\mathrm{e}}_{\mathrm{Route}})$ = IJmuiden

   $t_3(\in^{\mathrm{s}}_{\mathrm{Route}})$ = R_1
   $t_3(\in^{\mathrm{e}}_{\mathrm{Route}})$ = Tenerife

   Pop(I) = {1, 2}

t_1	1
t_2	2

This is an illegal population, because the third city in route R_1 does not have an index.

Constraint 7:

1. This constraint is supposed to specify the fact that each instance of the *Reviews* object type must play the role *are reviews of*. In other words this means that all reviews shown on TripAdvisor should be associated with at least one cruise.
2. total({are reviews of})
3. Suppose we have the following:
   Pop(Reviews) = {Very Disappointing, First Cruise}
   Pop(Cruise) = {7 Night Cruise to the Western Mediterranean,
   4 Night Cruise to Europe}

Very Disappointing	7 Night Cruise to the Western Mediterranean
Very Disappointing	4 Night Cruise to Europe

This is an illegal population, because the review First Cruise is not associated with any cruise.

Constraint 8:

1. This constraint is supposed to specify the fact that each instance of the *Cabins* object type must play the role *are cabins of*. In other words this means that all possible types of cabins shown on TripAdvisor should be associated with at least one cruise.
2. total({are cabins of})
3. Suppose we have the following:
   Pop(Cabins) = {Suite, Balcony}
   Pop(Cruise) = {7 Night Cruise to the Western Mediterranean,
   4 Night Cruise to Europe}

Suite	7 Night Cruise to the Western Mediterranean
Suite	4 Night Cruise to Europe

This is an illegal population, because the cabin type Balcony is not associated with any cruise.

Constraint 9:

1. This constraint is supposed to specify the fact that each instance of the *Cruise* object type must play either one of the roles *with* and *having* or both of them. In other words this means that all cruises shown on TripAdvisor should have reviews and/or deck information available.
2. total({having, with})
3. Suppose we have the following:
   Pop(Reviews) = {First Cruise}
   Pop(Decks) = {19}
   Pop(Cruise) = {7 Night Cruise to the Western Mediterranean,
   4 Night Cruise to Europe}

4 Night Cruise to Europe	First Cruise
4 Night Cruise to Europe	19

This is an illegal population, because the cruise 7 Night Cruise to the Western Mediterranean does not have any reviews and does not have any information available about the decks.

Constraint 10:

1. This constraint is supposed to specify the fact that each instance of the *Cruise* object type plays the role *has* at most once. In other words this means that each cruise shown on TripAdvisor should not have more than one route.
2. unique({has})
   identifier({has}, Itinerary)
3. Suppose we have the following:
   Pop(Cruise) = {4 Night Cruise to Europe}
   Pop(Route) = {R_1, R_2}
   R_1 = (Southampton, IJmuiden, IJmuiden, Southampton)
   R_2 = (Tenerife, Las Palmas, San Sebastian, Funchal, Lanzarote, Puerto del Rosario, Tenerife)

4 Night Cruise to Europe	R_1
4 Night Cruise to Europe	R_2

This is an illegal population, because the cruise 4 Night Cruise to Europe has more than one route.

Constraint 11:

1. This constraint is supposed to specify the fact that each instance of the *Route* sequence type should play the role $\in^{S}_{Route}$ at least twice and at most fourteen times. In other words this means that a route should contain at least two cities and at most fourteen cities.
2. frequency($\{\in^{S}_{Route}\}$, 2, 14)
   identifier($\{\in^{S}_{Route}\}$, $\in_{Route}$)
3. Suppose we have the following:
   Pop(City) = {IJmuiden}
   Pop(Route) = {R_1}
   R_1 = (IJmuiden)

R_1	IJmuiden

   This is an illegal population, because route R_1 does not contain at least 2 cities.

2.4.3 Population

$\text{Pop(Cruise)} = \{\text{7 Night Cruise to the Western Mediterranean, 4 Night Cruise to Europe}\}$

$\text{Pop(Cabins)} = \{\text{Suite, Balcony}\}$

$\text{Pop(Activities)} = \{\text{Gym, Casino, Lounge}\}$

$\text{Pop(Route)} = \{R_1, R_2\}$
$R_1 = \text{(Southampton, IJmuiden, IJmuiden, Southampton)}$
$R_2 = \text{(Tenerife, Las Palmas, San Sebastian, Funchal, Lanzarote, Puerto del Rosario, Tenerife)}$

$\text{Pop(City)} = \{\text{Tenerife, Las Palmas, San Sebastian, Funchal, Lanzarote,}$

$\text{Puerto del Rosario, Southampton, IJmuiden}\}$

$\text{Pop(Reviews)} = \{\text{Very Disapointing, First Cruise}\}$

$\text{Pop(Decks)} = \{13, 19\}$

$\text{Pop(Lodging)} = \{t_1, t_2, t_3\}$

$t_1\text{(are cabins of)} = \text{Suite}$

$t_1\text{(offering)} = \text{7 Night Cruise to the Western Mediterranean}$

$t_2\text{(are cabins of)} = \text{Balcony}$

$t_2\text{(offering)} = \text{7 Night Cruise to the Western Mediterranean}$

$t_3\text{(are cabins of)} = \text{Suite}$

$t_3\text{(offering)} = \text{4 Night Cruise to Europe}$

$\text{Pop(Entertainment)} = \{t_4, t_5, t_6, t_7\}$

t_4(is cruise offering) = 7 Night Cruise to the Western Mediterranean

t_4(on) = Lounge

t_5(is cruise offering) = 7 Night Cruise to the Western Mediterranean

t_5(on) = Gym

t_6(is cruise offering) = 4 Night Cruise to Europe

t_6(on) = Lounge

t_7(is cruise offering) = 4 Night Cruise to Europe

t_7(on) = Casino

$\text{Pop(Itinerary)} = \{t_8, t_9\}$

t_8(has) = 7 Night Cruise to the Western Mediterranean

t_8(of) = R_2

t_9(has) = 4 Night Cruise to Europe

t_9(of) = R_1

$\text{Pop}(\in_{\text{Route}}) = \{t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}\}$

$t_{10}(\in^{\text{s}}_{\text{Route}}) = R_1$

$t_{10}(\in^{\text{e}}_{\text{Route}})$ = Southampton

$t_{11}(\in^{\text{s}}_{\text{Route}}) = R_1$

$t_{11}(\in^{\text{e}}_{\text{Route}})$ = IJmuiden

$t_{12}(\in^{\text{s}}_{\text{Route}}) = R_2$

$t_{12}(\in^{\text{e}}_{\text{Route}})$ = Tenerife

$t_{13}(\in^{\text{s}}_{\text{Route}}) = R_2$

$t_{13}(\in^{\text{e}}_{\text{Route}})$ = Las Palmas

$t_{14}(\in^{\text{s}}_{\text{Route}}) = R_2$

$t_{14}(\in^{\text{e}}_{\text{Route}})$ = San Sebastian

$t_{15}(\in^{\text{s}}_{\text{Route}}) = R_2$

$t_{15}(\in^{\text{e}}_{\text{Route}})$ = Funchal

$t_{16}(\in^{\text{s}}_{\text{Route}}) = R_2$

$t_{16}(\in^{\text{e}}_{\text{Route}})$ = Lanzarote

$t_{17}(\in^{\text{s}}_{\text{Route}}) = R_2$

$t_{17}(\in^{\text{e}}_{\text{Route}})$ = Puerto del Rosario

$\mathrm{Pop}(@_{\mathrm{Route}}) = \{t_{18}, t_{19}, t_{20}, t_{21}, t_{22}, t_{23}, t_{24}, t_{25}, t_{26}, t_{27}, t_{28}\}$

$t_{18}(@^{\mathrm{s}}_{\mathrm{Route}}) = t_{10}$

$t_{18}(@^{\mathrm{i}}_{\mathrm{Route}}) = 1$

$t_{19}(@^{\mathrm{s}}_{\mathrm{Route}}) = t_{11}$

$t_{19}(@^{\mathrm{i}}_{\mathrm{Route}}) = 2$

$t_{20}(@^{\mathrm{s}}_{\mathrm{Route}}) = t_{11}$

$t_{20}(@^{\mathrm{i}}_{\mathrm{Route}}) = 3$

$t_{21}(@^{\mathrm{s}}_{\mathrm{Route}}) = t_{10}$

$t_{21}(@^{\mathrm{i}}_{\mathrm{Route}}) = 4$

$t_{22}(@^{\mathrm{s}}_{\mathrm{Route}}) = t_{12}$

$t_{22}(@^{\mathrm{i}}_{\mathrm{Route}}) = 1$

$t_{23}(@^{\mathrm{s}}_{\mathrm{Route}}) = t_{13}$

$t_{23}(@^{\mathrm{i}}_{\mathrm{Route}}) = 2$

$t_{24}(@^{\mathrm{s}}_{\mathrm{Route}}) = t_{14}$

$t_{24}(@^{\mathrm{i}}_{\mathrm{Route}}) = 3$

$t_{25}(@^{\mathrm{s}}_{\mathrm{Route}}) = t_{15}$

$t_{25}(@^{\mathrm{i}}_{\mathrm{Route}}) = 4$

$t_{26}(@^{\mathrm{s}}_{\mathrm{Route}}) = t_{16}$

$t_{26}(@^{\mathrm{i}}_{\mathrm{Route}}) = 5$

$t_{27}(@^{\mathrm{s}}_{\mathrm{Route}}) = t_{17}$

$t_{27}(@^{\mathrm{i}}_{\mathrm{Route}}) = 6$

$t_{28}(@^{\mathrm{s}}_{\mathrm{Route}}) = t_{12}$

$t_{28}(@^{\mathrm{i}}_{\mathrm{Route}}) = 7$

$\mathrm{Pop}(\mathrm{Rating}) = \{t_{29}, t_{30}\}$

t_{29}(with) = 7 Night Cruise to the Western Mediterranean

t_{29}(are reviews of) = First Cruise

t_{30}(with) = 4 Night Cruise to Europe

t_{30}(are reviews of) = Very Disappointing

$\mathrm{Pop}(\mathrm{Size}) = \{t_{31}, t_{32}\}$

t_{31}(are decks on) = 19

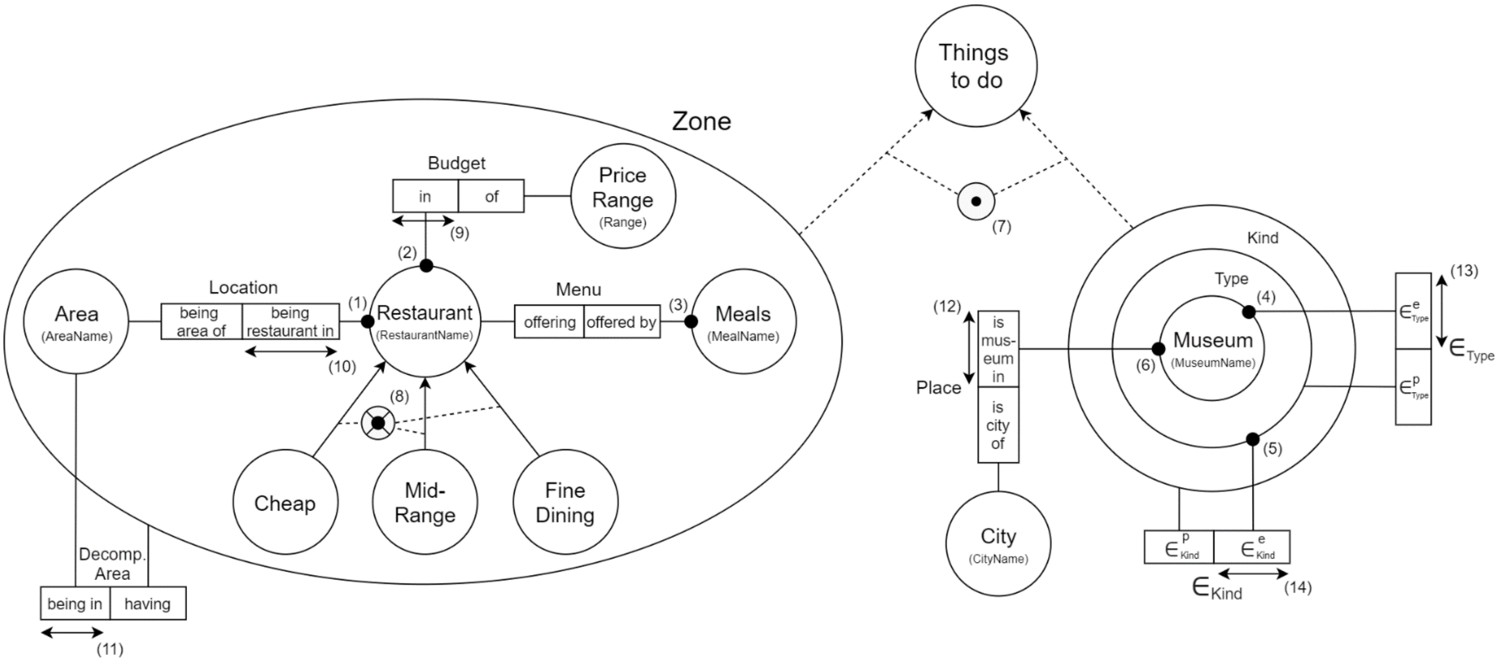
t_{31}(having) = 7 Night Cruise to the Western Mediterranean

t_{32}(are decks on) = 13

t_{32}(having) = 4 Night Cruise to Europe

2.5 Fragment of things to do

2.5.1 Diagram



2.5.2 Formal definition

$\mathcal{E} = \{\text{Area, Price Range, Meals, Fine Dining, Mid-Range, Cheap, Things to do, Museum, City}\}$

$\text{Spec} \subseteq \mathcal{E} \times \mathcal{O}$

Cheap Spec Restaurant

Mid-Range Spec Restaurant

Fine Dining Spec Restaurant

We defined a subtype defining rule for our specialization in the following way:
Cheap = All restaurants with a price range ≤ 15.
Mid-Range = All restaurants with a price range 15 < price range ≤ 30.
Fine-Dining = All restaurants with a price range > 30.

$\text{Gen} \subseteq \mathcal{E} \times \mathcal{O}$

Things to do Gen Zone

Things to do Gen City

$\mathcal{F} = \{\text{Budget, Menu, Location, Decomp. Area}, \in_{\text{Type}}, \in_{\text{Kind}}, \text{Place}\}$

$\mathcal{P} = \{\text{in, of, offering, offered by, being restaurant in, being area of, being in, having},$

$\in^{p}_{\text{Kind}}, \in^{e}_{\text{Kind}}, \in^{p}_{\text{Type}}, \in^{e}_{\text{Type}}, \text{is museum in, is city of}\}$

$\mathcal{G} = \{\text{Kind, Type}\}$

$\mathcal{C} = \{\text{Zone}\}$

$\mathcal{L} = \{\text{AreaName, Range, MealName, MuseumName, CityName}\}$

$\mathcal{O} = \mathcal{E} \cup \mathcal{F} \cup \mathcal{G} \cup \mathcal{C} \cup \mathcal{L}$

$\mathrm{Location} = \{\text{being area of, being restaurant in}\}$

$\mathrm{Budget} = \{\text{in, of}\}$

$\mathrm{Menu} = \{\text{offering, offered by}\}$

$\mathrm{Decomp.\,Area} = \{\text{being in, having}\}$

$\in_{\mathrm{Type}} = \left\{\in^{\mathrm{p}}_{\mathrm{Type}}, \in^{\mathrm{e}}_{\mathrm{Type}}\right\}$

$\in_{\mathrm{Kind}} = \{\in^{\mathrm{p}}_{\mathrm{Kind}}, \in^{\mathrm{e}}_{\mathrm{Kind}}\}$

$\mathrm{City} = \{\text{is museum in, is city of}\}$

$\mathrm{Base}(\text{in}) = \mathrm{Base}(\text{offering}) = \mathrm{Base}(\text{being restaurant in}) = \mathrm{Restaurant}$

$\mathrm{Base}(\text{of}) = \text{Price Range}$

$\mathrm{Base}(\text{offered by}) = \mathrm{Meals}$

$\mathrm{Base}(\text{being area of}) = \mathrm{Base}(\text{being in}) = \mathrm{Area}$

$\mathrm{Base}(\text{having}) = \mathrm{Zone}$

$\mathrm{Base}\left(\in^{\mathrm{p}}_{\mathrm{Kind}}\right) = \mathrm{Kind}$

$\mathrm{Base}(\in^{\mathrm{e}}_{\mathrm{Kind}}) = \mathrm{Base}\left(\in^{\mathrm{p}}_{\mathrm{Type}}\right) = \mathrm{Type}$

$\mathrm{Base}\left(\in^{\mathrm{e}}_{\mathrm{Type}}\right) = \mathrm{Base}(\text{is museum in}) = \mathrm{Museum}$

$\mathrm{Base}(\text{is city of}) = \mathrm{City}$

$\mathrm{Elt}(\mathrm{Type}) = \mathrm{Museum}$

$\mathrm{Elt}(\mathrm{Kind}) = \mathrm{Type}$

$\mathrm{Zone} \prec \mathrm{Restaurant}$

$\mathrm{Zone} \prec \mathrm{Area}$

$\mathrm{Zone} \prec \text{Price Range}$

$\mathrm{Zone} \prec \mathrm{Meals}$

$\mathrm{Zone} \prec \text{Fine Dining}$

$\mathrm{Zone} \prec \text{Mid-Range}$

$\mathrm{Zone} \prec \mathrm{Cheap}$

$\mathrm{Zone} \prec \mathrm{Location}$

$\mathrm{Zone} \prec \mathrm{Budget}$

$\mathrm{Zone} \prec \mathrm{Menu}$

Formal definition of the constraints:

First we show type relatedness within our model:
Restaurant ~ Cheap ~ Mid-Range ~ Fine Dining
being restaurant in ~ in ~ offering
Zone ~ Things to do ~ Kind
Museum ~ Type ~ Kind

In order to specify the formal definition of the constraints of this model we added numbers to the constraints in the drawing such that we can refer to them and we take the following steps for every constraint:

1. Explain informally what the constraint is supposed to do.
2. Give the formal semantics.
3. Create an illegal population.

Constraint 1:

1. This constraint is supposed to specify the fact that each instance of the *Restaurant* object type must play the role *being restaurant in*. In other words this means that all restaurants shown on TripAdvisor should be in at least one area.
2. total({being restaurant in})
3. Suppose we have the following:
   Pop(Restaurant) = {Happies, Neso}
   Pop(Area) = {Europe}

Happies	Europe

   This is an illegal population, because restaurant Neso is not in any area.

Constraint 2:

1. This constraint is supposed to specify the fact that each instance of the *Restaurant* object type must play the role *in*. In other words this means that all restaurants shown on TripAdvisor should have at least one price range.
2. total({in})
3. Suppose we have the following:
   Pop(Restaurant) = {Happies, Neso}
   Pop(Price Range) = {4-6, 81-108}

Happies	4-6

   This is an illegal population, because restaurant Neso is not associated with any price range.

Constraint 3:

1. This constraint is supposed to specify the fact that each instance of the *Meals* object type must play the role *offered by*. In other words this means that all meals shown on TripAdvisor should be offered by at least one restaurant.
2. total({offered by})
3. Suppose we have the following:
   Pop(Meals) = {Lobster, Pressed stew}
   Pop(Restaurant) = {Happies, Bistrot Orcia}

Lobster	Happies
Lobster	Bistrot Orcia

   This is an illegal population, because the meal pressed stew is not offered by any restaurant.

Constraint 4:

1. This constraint is supposed to specify the fact that each instance of the *Museum* object type must play the role $\in^{e}_{Type}$. In other words this means that all museums shown on TripAdvisor should be of at least one type.
2. total($\{\in^{\mathrm{e}}_{\mathrm{Type}}\}$)
3. Suppose we have the following:
   Pop(Museum) = {Museum of Art, Museum of Pop music}
   Pop(Type) = {Art} where
   Art = {Museum of Art}

Museum of Art	Art

   This is an illegal population, because Museum of Pop music is not of any type.

Constraint 5:

1. This constraint is supposed to specify the fact that each instance of the *Type* power type must play the role $\in^{e}_{Kind}$. In other words this means that all types shown on TripAdvisor should be of at least one kind.
2. total($\{\in^{\mathrm{e}}_{\mathrm{Kind}}\}$)
3. Suppose we have the following:
   Pop(Type) = {Art, Music} where
   Art = {Museum of Art}
   Music = {Museum of Pop music}
   Pop(Kind) = {Culture} where
   Culture = {Art}

Art	Culture

   This is an illegal population, because the type Music is not of any kind.

Constraint 6:

1. This constraint is supposed to specify the fact that each instance of the *Museum* object type must play the role *is museum in*. In other words this means that all museums shown on TripAdvisor should be in at least one city.
2. total({is museum in})
3. Suppose we have the following:
   Pop(Museum) = {Museum of Art, Natural History Museum}
   Pop(City) = {Paris}

Museum of Art	Paris

   This is an illegal population, because the Natural History Museum does not have a city in which it is located.

Constraint 7:

1. This constraint is supposed to specify the fact that each instance of the *Things to do* object type must be of type *Zone* or of type *Kind* or of both types. In other words this means that all things to do shown on TripAdvisor should be restaurants and/or museums.
2. sub_total({Things to do})
3. Suppose we have the following:
   Pop(Things to do) = {Pop_{Europe}, Culture, History}
   Pop(Zone) = {Pop_{Europe}}
   Pop(Kind) = {Culture}
   This is an illegal population, because History is neither in the population of Zone or the population of Kind and thus Pop(Zone) ∪ Pop(Kind) ≠ Pop(Things to do).

Constraint 8:

1. This constraint basically consists of 2 constraints, the total role constraint and the exclusion constraint. The total role constraint is supposed to specify the fact that each instance of the *Restaurant* object type must be one of the three specializations or two of those specializations or all of them. Moreover, the exclusion constraint is supposed to specify the fact that an instance of the *Restaurant* object type can only be one of the three specializations. In other words, the combination of these constraints means that all restaurants shown on TripAdvisor should have exactly one price type.
2. sub_total({Restaurant})
   sub_exclusion({Restaurant})
3. Suppose we have the following:
   Pop(Restaurant) = {Happies, Neso, Bistrot Orcia}
   Pop(Cheap) = {Happies, Neso}
   Pop(Mid-Range) = {Happies}
   This is an illegal population, because we violate both the total role constraint and the exclusion constraint. Restaurant Happies has two price types, while the restaurant Bistrot Orcia does not have any price type.

Constraint 9:

1. This constraint is supposed to specify the fact that each instance of the *Restaurant* object type plays the role *in* at most once. In other words this means that each restaurant shown on TripAdvisor should not have more than one price range.
2. unique({in})
   identifier({in}, Budget)
3. Suppose we have the following:
   Pop(Restaurant) = {Happies}
   Pop(Price Range) = {4-6, 81-108}

Happies	4-6
Happies	81-108

This is an illegal population, because the restaurant Happies has more than one price range.

Constraint 10:

1. This constraint is supposed to specify the fact that each instance of the *Restaurant* object type plays the role *being restaurant in* at most once. In other words this means that each restaurant shown on TripAdvisor should be in at most one area.
2. unique({being restaurant in})
   identifier({being restaurant in}, Location)
3. Suppose we have the following:
   Pop(Restaurant) = {Happies}
   Pop(Area) = {Europe, North America}

Happies	Europe
Happies	North America

   This is an illegal population, because the restaurant Happies is in more than one area.

Constraint 11:

1. This constraint is supposed to specify the fact that each instance of the *Area* object type plays the role *being in* at most once. In other words this means that each area shown on TripAdvisor can be decomposed at most once.
2. unique({being in})
   identifier({being in}, Decomp. Area)
3. Suppose we have the following:
   Pop(Area) = {Europe}
   Pop(Zone) = {Pop_{Europe}, $Pop_{North\ America}$}

Europe	Pop_{Europe}
Europe	$Pop_{North\ America}$

   This is an illegal population, because the area Europe is decomposed more than once.

Constraint 12:

1. This constraint is supposed to specify the fact that each instance of the *Museum* object type plays the role *is museum in* at most once. In other words this means that each museum shown on TripAdvisor should be in at most one city.
2. unique({is museum in})
   identifier({is museum in}, Place)
3. Suppose we have the following:
   Pop(Museum) = {Natural History Museum}
   Pop(City) = {Paris, Cluj}

Natural History Museum	Paris
Natural History Museum	Cluj

   This is an illegal population, because the Natural History Museum is located in more than one city.

Constraint 13:

1. This constraint is supposed to specify the fact that each instance of the *Museum* object type plays the role $\in^{e}_{Type}$ at most once. In other words this means that each museum shown on TripAdvisor should be of at most one type.
2. unique($\{\in^{\mathrm{e}}_{\mathrm{Type}}\}$)
   identifier($\{\in^{\mathrm{e}}_{\mathrm{Type}}\}, \in_{\mathrm{Type}}$)
3. Suppose we have the following:
   Pop(Museum) = {Louvre}
   Pop(Type) = {Art, Music} where
   Art = {Louvre}
   Music = {Louvre}

Louvre	Art
Louvre	Music

   This is an illegal population, because the museum Louvre is of more than one type.

Constraint 14:

1. This constraint is supposed to specify the fact that each instance of the *Type* power type plays the role $\in^{e}_{Kind}$ at most once. In other words this means that each type shown on TripAdvisor should be of at most one kind.
2. unique($\{\in^{\mathrm{e}}_{\mathrm{Kind}}\}$)
   identifier($\{\in^{\mathrm{e}}_{\mathrm{Kind}}\}, \in_{\mathrm{Kind}}$)
3. Suppose we have the following:
   Pop(Type) = {Art} where
   Art = {Louvre}
   Pop(Kind) = {History, Culture} where
   History = {Art}
   Culture = {Art}

Art	History
Art	Culture

   This is an illegal population, because the type Art is of more than one kind.

2.5.3 Population

$\text{Pop(Things to do)} = \text{Pop(Zone)} \cup \text{Pop(Kind)}$

$\text{Pop(City)} = \{\text{Paris, Cluj}\}$

$\text{Pop(Kind)} = \{\text{History, Culture}\}$

Where $\text{History} = \{\text{Natural History, Transylvanian History}\}$
$\text{Culture} = \{\text{Art, Music}\}$

$\text{Pop(Type)} = \{\text{Transylvanian History, Natural History, Art, Music}\}$

Where $\text{Art} = \{\text{Louvre, Museum of Art}\}$
$\text{Music} = \{\text{Museum of Pop music}\}$
$\text{Natural History} = \{\text{Natural History Museum}\}$
$\text{Transylvanian History} = \{\text{Transylvanian History Museum}\}$

$\text{Pop(Museum)} = \{\text{Transylvanian History Museum, Natural History Museum, Louvre,}$
$\text{Museum of Art, Museum of Pop music}\}$

$\text{Pop(Place)} = \{t_1, t_2, t_3, t_4, t_5\}$

$t_1(\text{is museum in}) = \text{Transylvanian History Museum}$

$t_1(\text{is city of}) = \text{Cluj}$

$t_2(\text{is museum in}) = \text{Natural History Museum}$

$t_2(\text{is city of}) = \text{Cluj}$

$t_3(\text{is museum in}) = \text{Louvre}$

$t_3(\text{is city of}) = \text{Paris}$

$t_4(\text{is museum in}) = \text{Museum of Art}$

$t_4(\text{is city of}) = \text{Paris}$

$t_5(\text{is museum in}) = \text{Museum of Pop music}$

$t_5(\text{is city of}) = \text{Paris}$

$\text{Pop}(\in_{\text{Type}}) = \{t_6, t_7, t_8, t_9, \mathrm{t}_{10}\}$

$t_6(\in^{\text{e}}_{\text{Type}}) = \text{Transylvanian History Museum}$

$t_6(\in^{\text{p}}_{\text{Type}}) = \text{Transylvanian History}$

$t_7(\in^{\text{e}}_{\text{Type}}) = \text{Natural History Museum}$

$t_7(\in^{\text{p}}_{\text{Type}}) = \text{Natural History}$

$t_8(\in^{\text{e}}_{\text{Type}}) = \text{Louvre}$

$t_8(\in^{\text{p}}_{\text{Type}}) = \text{Art}$

$t_9(\in^{\text{e}}_{\text{Type}}) = \text{Museum of Art}$

$t_9(\in^{\text{p}}_{\text{Type}}) = \text{Art}$

$t_{10}(\in^{\text{e}}_{\text{Type}}) = \text{Museum of Pop music}$

$t_{10}(\in^{\text{p}}_{\text{Type}}) = \text{Music}$

$\mathrm{Pop}(\in_{\mathrm{Kind}}) = \{t_{11}, t_{12}, t_{13}, \mathrm{t}_{14}\}$

$t_{11}(\in^{\mathrm{e}}_{\mathrm{Kind}}) =$ Transylvanian History

$t_{11}(\in^{\mathrm{p}}_{\mathrm{Kind}}) =$ History

$t_{12}(\in^{\mathrm{e}}_{\mathrm{Kind}}) =$ Natural History

$t_{12}(\in^{\mathrm{p}}_{\mathrm{Kind}}) =$ History

$t_{13}(\in^{\mathrm{e}}_{\mathrm{Kind}}) =$ Art

$t_{13}(\in^{\mathrm{p}}_{\mathrm{Kind}}) =$ Culture

$t_{14}(\in^{\mathrm{e}}_{\mathrm{Kind}}) =$ Music

$t_{14}(\in^{\mathrm{p}}_{\mathrm{Kind}}) =$ Culture

We will now specify the population of our schema type:

$\mathrm{Pop} \in \mathrm{Pop}_i$ $\quad \mathrm{Pop}_{\mathrm{Europe}} \in \mathrm{Pop}_j$ $\quad \mathrm{Pop}_{\mathrm{North\ America}} \in \mathrm{Pop}_j$

$\mathrm{Pop}_{\mathrm{Germany}} \in \mathrm{Pop}_j$ $\quad \mathrm{Pop}_{\mathrm{France}} \in \mathrm{Pop}_j$

$\mathrm{Pop}(\mathrm{Restaurant}) = \{\mathrm{Happies, Neso, Bistrot\ Orcia}\}$

$\mathrm{Pop}(\mathrm{Price\ Range}) = \{4\text{-}6, 81\text{-}108, 16\text{-}27\}$

$\mathrm{Pop}(\mathrm{Meals}) = \{\mathrm{Klara, Lobster, Pressed\ stew}\}$

$\mathrm{Pop}(\mathrm{Area}) = \{\mathrm{Europe, North\ America, France, Germany, Paris, Lyon, Berlin}\}$

$\mathrm{Pop}(\mathrm{Cheap}) = \{\mathrm{Happies}\}$

$\mathrm{Pop}(\mathrm{Mid\text{-}Range}) = \{\mathrm{Bistrot\ Orcia}\}$

$\mathrm{Pop}(\mathrm{Fine\ Dining}) = \{\mathrm{Neso}\}$

$\mathrm{Pop}(\mathrm{Location}) = \{t_{15}, t_{16}, \mathrm{t}_{17}\}$

$t_{15}(\mathrm{being\ area\ of}) =$ Europe

$t_{15}(\mathrm{being\ restaurant\ in}) =$ Happies

$t_{16}(\mathrm{being\ area\ of}) =$ Europe

$t_{16}(\mathrm{being\ restaurant\ in}) =$ Bistrot Orcia

$t_{17}(\mathrm{being\ area\ of}) =$ Europe

$t_{17}(\mathrm{being\ restaurant\ in}) =$ Neso

$\mathrm{Pop}(\mathrm{Budget}) = \{t_{18}, t_{19}, \mathrm{t}_{20}\}$

$t_{18}(\mathrm{in}) =$ Happies

$t_{18}(\mathrm{of}) =$ 4-6

$t_{19}(\mathrm{in}) =$ Bistrot Orcia

$t_{19}(\mathrm{of}) =$ 16-27

$t_{20}(\mathrm{in}) =$ Neso

$t_{20}(\mathrm{of}) =$ 81-108

$\mathrm{Pop}(\mathrm{Menu}) = \{t_{21}, t_{22}, t_{23}\}$

$t_{21}(\text{offering}) = \text{Happies}$

$t_{21}(\text{offered by}) = \text{Klara}$

$t_{22}(\text{offering}) = \text{Bistrot Orcia}$

$t_{22}(\text{offered by}) = \text{Pressed stew}$

$t_{23}(\text{offering}) = \text{Neso}$

$t_{23}(\text{offered by}) = \text{Lobster}$

$\mathrm{Pop}(\text{Decomp. Area}) = \{t_{24}, t_{25}\}$

$t_{24}(\text{being in}) = \text{Europe}$

$t_{24}(\text{having}) = \mathrm{Pop}_{\text{Europe}}$

$t_{25}(\text{being in}) = \text{North America}$

$t_{25}(\text{having}) = \mathrm{Pop}_{\text{North America}}$

Since $\mathrm{Pop}_{\text{North America}}$ in our decomposition is actually empty, we decided to not specify it any further here. We continue with specifying what $\mathrm{Pop}_{\text{Europe}}$ contains.

$\mathrm{Pop}_{\text{Europe}}(\text{Restaurant}) = \{\text{Happies, Neso, Bistrot Orcia}\}$

$\mathrm{Pop}_{\text{Europe}}(\text{Price Range}) = \{\text{4-6, 81-108, 16-27}\}$

$\mathrm{Pop}_{\text{Europe}}(\text{Meals}) = \{\text{Klara, Lobster, Pressed stew}\}$

$\mathrm{Pop}_{\text{Europe}}(\text{Area}) = \{\text{France, Germany, Paris, Lyon, Berlin}\}$

$\mathrm{Pop}_{\text{Europe}}(\text{Cheap}) = \{\text{Happies}\}$

$\mathrm{Pop}_{\text{Europe}}(\text{Mid-Range}) = \{\text{Bistrot Orcia}\}$

$\mathrm{Pop}_{\text{Europe}}(\text{Fine Dining}) = \{\text{Neso}\}$

$\mathrm{Pop}_{\text{Europe}}(\text{Location}) = \{t_{26}, t_{27}, t_{28}\}$

$t_{26}(\text{being area of}) = \text{Germany}$

$t_{26}(\text{being restaurant in}) = \text{Happies}$

$t_{27}(\text{being area of}) = \text{France}$

$t_{27}(\text{being restaurant in}) = \text{Bistrot Orcia}$

$t_{28}(\text{being area of}) = \text{France}$

$t_{28}(\text{being restaurant in}) = \text{Neso}$

$\mathrm{Pop}_{\text{Europe}}(\text{Budget}) = \{t_{29}, t_{30}, t_{31}\}$

$t_{29}(\text{in}) = \text{Happies}$

$t_{29}(\text{of}) = \text{4-6}$

$t_{30}(\text{in}) = \text{Bistrot Orcia}$

$t_{30}(\text{of}) = \text{16-27}$

$t_{31}(\text{in}) = \text{Neso}$

$t_{31}(\text{of}) = \text{81-108}$

$\mathrm{Pop}_{\mathrm{Europe}}(\mathrm{Menu}) = \{t_{32}, t_{33}, t_{34}\}$

t_{32}(offering) = Happies

t_{32}(offered by) = Klara

t_{33}(offering) = Bistrot Orcia

t_{33}(offered by) = Pressed stew

t_{34}(offering) = Neso

t_{34}(offered by) = Lobster

$\mathrm{Pop}_{\mathrm{Europe}}(\mathrm{Decomp.\ Area}) = \{t_{35}, t_{36}\}$

t_{35}(being in) = Germany

t_{35}(having) = $\mathrm{Pop}_{\mathrm{Germany}}$

t_{36}(being in) = France

t_{36}(having) = $\mathrm{Pop}_{\mathrm{France}}$

$\mathrm{Pop}_{\mathrm{Germany}}(\mathrm{Restaurant}) = \{\mathrm{Happies}\}$

$\mathrm{Pop}_{\mathrm{Germany}}(\mathrm{Price\ Range}) = \{4\text{-}6\}$

$\mathrm{Pop}_{\mathrm{Germany}}(\mathrm{Meals}) = \{\mathrm{Klara}\}$

$\mathrm{Pop}_{\mathrm{Germany}}(\mathrm{Area}) = \{\mathrm{Berlin}\}$

$\mathrm{Pop}_{\mathrm{Germany}}(\mathrm{Cheap}) = \{\mathrm{Happies}\}$

$\mathrm{Pop}_{\mathrm{Germany}}(\mathrm{Mid\text{-}Range}) = \emptyset$

$\mathrm{Pop}_{\mathrm{Germany}}(\mathrm{Fine\ Dining}) = \emptyset$

$\mathrm{Pop}_{\mathrm{Germany}}(\mathrm{Location}) = \{t_{37}\}$

t_{37}(being area of) = Berlin

t_{37}(being restaurant in) = Happies

$\mathrm{Pop}_{\mathrm{Germany}}(\mathrm{Budget}) = \{t_{38}\}$

t_{38}(in) = Happies

t_{38}(of) = 4-6

$\mathrm{Pop}_{\mathrm{Germany}}(\mathrm{Menu}) = \{t_{39}\}$

t_{39}(offering) = Happies

t_{39}(offered by) = Klara

$\mathrm{Pop}_{\mathrm{Germany}}(\mathrm{Decomp.\ Area}) = \emptyset$

$\mathrm{Pop}_{\mathrm{France}}(\text{Restaurant}) = \{\text{Neso, Bistrot Orcia}\}$

$\mathrm{Pop}_{\mathrm{France}}(\text{Price Range}) = \{\text{81-108, 16-27}\}$

$\mathrm{Pop}_{\mathrm{France}}(\text{Meals}) = \{\text{Lobster, Pressed stew}\}$

$\mathrm{Pop}_{\mathrm{France}}(\text{Area}) = \{\text{Paris}, \text{Lyon}\}$

$\mathrm{Pop}_{\mathrm{France}}(\text{Cheap}) = \emptyset$

$\mathrm{Pop}_{\mathrm{France}}(\text{Mid-Range}) = \{\text{Bistrot Orcia}\}$

$\mathrm{Pop}_{\mathrm{France}}(\text{Fine Dining}) = \{\text{Neso}\}$

$\mathrm{Pop}_{\mathrm{France}}(\text{Location}) = \{t_{40}, t_{41}\}$

$t_{40}(\text{being area of}) = \text{Paris}$

$t_{40}(\text{being restaurant in}) = \text{Neso}$

$t_{41}(\text{being area of}) = \text{Lyon}$

$t_{41}(\text{being restaurant in}) = \text{Bistrot Orcia}$

$\mathrm{Pop}_{\mathrm{France}}(\text{Budget}) = \{t_{42}, t_{43}\}$

$t_{42}(\text{in}) = \text{Bistrot Orcia}$

$t_{42}(\text{of}) = \text{16-27}$

$t_{43}(\text{in}) = \text{Neso}$

$t_{43}(\text{of}) = \text{81-108}$

$\mathrm{Pop}_{\mathrm{France}}(\text{Menu}) = \{t_{44}, t_{45}\}$

$t_{44}(\text{offering}) = \text{Bistrot Orcia}$

$t_{44}(\text{offered by}) = \text{Pressed stew}$

$t_{45}(\text{offering}) = \text{Neso}$

$t_{45}(\text{offered by}) = \text{Lobster}$

$\mathrm{Pop}_{\mathrm{France}}(\text{Decomp. Area}) = \emptyset$

$\mathrm{Pop}(\text{Zone}) = \{\mathrm{Pop}_{\mathrm{Europe}}, \mathrm{Pop}_{\mathrm{North\ America}}, \mathrm{Pop}_{\mathrm{Germany}}, \mathrm{Pop}_{\mathrm{France}}\}$

Chapter 3 – Part B

3.1 Introduction

Throughout the process of creating models and their mathematical formalizations we discovered that while the theory is complex enough to offer many possibilities, it also has certain drawbacks that makes it harder for us to come up with models that are as close as possible to the applications that exist in the real world. In this chapter, we will evaluate the applicability of the theory in terms of strengths and weaknesses, based on our application in Chapter 2.

3.1 Strengths and weaknesses

One of the first weaknesses that we discovered is the fact that the given definition of the power type is pretty restrictive. In our original idea for the fourth model, we tried to first organize the museums based on their location with a power type for *City*. Then we tried to create a power type of a power type to organize those museums again also based on their type. However, then we realised that this is not possible since a city can have museums of different types, but the power type notation does not allow a set of a city to be broken apart on types since it will violate what the power type is supposed to do. We could only fix this by adding the fact type *Place* and by changing the way we want to organize the museums with power types.
A second weakness that we found is that it happens to be really difficult to specify a big population that might make a model easier to understand. We often had to reduce ourselves to a few instances for each entity type and even then sometimes our populations would get really big.
The third weakness that we want to mention is the fact that the possibilities for the naming of the predicators are really limited. We always had to make sure we did not have two predicators with the same name and it was also difficult to come up with names that make sense for our model. Moreover, there is not really a good way to name the predicators of ternary or k-nary fact types such that the reading of them makes sense in every direction.
The fourth weakness that we discovered is the absence of a set constraint that only allows certain values for an object type. For example, in our model fragment of the Cruise information system, we wanted to model the deck numbers such that we only allow numbers from 5 upwards, because in a real-life situation the lower decks are not available for guests. This a different from a frequency constraint that does exist in the theory.
Finally, we had to add numbering to the drawings of our models in order to refer to them and to be able to explain what each constraint does in a clear way. This addition is not part of the theory so we are going to consider this a weakness.

Now we are going to discuss the strengths of the theory that we found.
The first strength that we discovered is the amount of possibilities the current theory offers to represent correct populations that are close to real situations. In all of our models we were able to use constraints such that our formalization is really close to what the TripAdvisor website does in that specific case. Therefore, we think that the theory that we were allowed to use and that got explained is already powerful enough to formalize many real-life applications, which is useful for a computer scientist.

A second strength that we found is the fact that in many cases a drawing of a model is not needed anymore, since the mathematical definitions are clear and advanced enough to understand what a model is supposed to represent. These definitions make sure that models will be less confusing and more straightforward.
Finally, a third strength that we found is the schema type. It is a really handy piece of theory that helped us in representing a really large and complex model in an understandable way. The schema type provides many advanced applications and our use of it is just one of them.

Chapter 4 – Part C

4.1 Introduction

In this chapter, we are going to propose and specify possible extensions of the theory, in order to improve the applicability. We are going to remove the weaknesses that we described in the previous chapter.

4.2 Extensions and improvements

In order to fix the first weakness that we found in the previous chapter - the weakness of the power types – it is possible to change the theory such that it is allowed to break those sets apart for further power types. Basically, if we would have used a power type to organize the museums based on their type for our fourth model, then if we would change the theory in the way we would need it for using a power type to also organize them on cities, the requirement that a power type should be a subset of the type needs to be removed. This will allow the second power type to break apart the sets of the first power type and thus this creates a less restrictive way of using power types in models.
To fix the second weakness – the weakness of big populations – it is possible to allow big populations for the object types, but not require to make all the possible combinations for the fact types.
In order to solve our third weakness that we found in chapter 3, we thought of allowing predicators to not be just names but also sets names. In this way, it is possible to read the predicators nicely in more than one direction and also the naming gets more unique. A more unique naming provides less naming conflicts.
The fourth weakness discussed in the previous chapter can be solved by adding the set constraint that we described.
Our last weakness described in chapter 3 is a weakness that we already solved by adding a numbering to the drawings of the models.