Constraints Extraction Out Of Airplane Fare Rules

Madhur Malviya,
Indian Institute of Technology,
Guwahati,
India.

Jean-Cédric Chappelier,
École Polytechnique Fédérale de Lausanne,
Switzerland.

July 2000
# Table of Contents

1 Acknowledgements ....................................................................................................................... 1  
2 Introduction................................................................................................................................... 2  
3 SYSTEM ARCHITECTURE ........................................................................................................... 3  
   3.1 PRE-PROCESSOR ................................................................................................................ 3  
      The Architecture of the System ............................................................................................ 4  
      3.1.1 INITIAL PROCESSING OF INPUT DATA ................................................................. 5  
      3.1.2 TAGGER .................................................................................................................... 5  
      3.1.3 TAG REMOVER ........................................................................................................ 5  
   3.2 LEXICAL ANALYSER .......................................................................................................... 6  
      3.2.1 TOKENIZER ............................................................................................................... 6  
      3.2.2 COUNTERS ............................................................................................................... 7  
   3.3 SYNTACTIC ANALYSIS ...................................................................................................... 7  
      3.3.1 SENTENCE ALIGNER .............................................................................................. 7  
      3.3.2 DEVELOPMENT PHASE ......................................................................................... 8  
      3.3.3 EXPLOITATION PHASE........................................................................................... 8  
   3.4 CONSTRAINT EXTRACTOR .............................................................................................. 9  
   3.5 AN EXAMPLE .................................................................................................................... 10  
      3.5.1 The constraints generated: ........................................................................................ 10  
      3.5.2 The original file taken from the web (page 1): ......................................................... 11  
      3.5.3 The file after pre-processing (page 1):................................................................. 12  
      3.5.4 The original file taken from the web (page 2): ......................................................... 13  
      3.5.5 The file after pre-processing (page 2):................................................................. 14  
4 CONCLUSION ........................................................................................................................... 15  
5 FUTURE WORK ........................................................................................................................ 15  
6 APPENDIX .................................................................................................................................. 16  
   6.1 Variables in the Constraints: ............................................................................................ 16  
   6.2 The Tag Set ....................................................................................................................... 17  
   6.3 Grammar ............................................................................................................................ 18  
   6.4 Software Organization ......................................................................................................... 22
Abstract
The purpose of this project is to analyze the flight rules specified by airline companies and generate constraints out of them.
The first phase concentrates on building up the resources. Downloaded data from the Internet is analyzed to build the lexicon and the grammar mandatory for the analysis of the content.
In the exploitation phase, the data is parsed to look for and structure the occurrence of important sentences. The constraint extractor uses the output of the parser to find sentences which could generate constraints. Once such a sentence is detected, the corresponding constraint is reformulated as output.
1 Acknowledgements

I express my heartfelt gratitude to Prof Boi Faltings for his help and support, and also for giving me the opportunity to work under his guidance.
I am highly grateful to my guide Dr Jean-Cédric Chappelier for the unstinted help and kind co-operation he provided me with.
2 Introduction

In the current technical scenario, a large portion of the databases that exist are being put on the Internet. One example is the database of airline and flight related information. An air travel planner would offer all facilities that a travel agent offers today, better still, all these facilities will available in an interactive form over the Internet.

Since the airline companies have to provide their rules in “natural language” for legal reasons, this form is considered as the input form for this project.

Each airline company specifies a set of fare rules for every flight it offers. These rules are dynamic, and are different for different routes, fares, classes etc. The objective of this project is to translate the set of rules into constraints which can then be solved using the various methods for solving CSPs.

The databases of flight ticket availability are used only as repositories of data, while all the work is done by human operators. These databases can be used by computer programs to book tickets for prospective passengers online as well. It would be very convenient if the reservation of tickets for a flight were done directly via computers without the need for an operator.

This involves contributions from various fields of Computer Science and Engineering. One such important contribution is from the area of Artificial Intelligence, which helps to accomplish the important task of making the selected itinerary conform to the conditions specified by both the passenger and the airline company. This work was done during my summer project at the École Polytechnique Fédérale de Lausanne.

The project work consists of two major phases:

- The Resource Development phase concentrates on developing the data needed for successful parsing of the fare rules. This includes writing the lexicon and the grammar from manual analysis of samples.
- The Exploitation phase does the job of using the resources to generate the constraints. This phase is fully automatic.

This report goes through all the phases of the project. The higher level modules are introduced first, and then the sub-modules are explained. Some of the work was done only for the purpose of analysis of the input text and not used during the exploitation phase. This has clearly been demarcated in the report.
3 SYSTEM ARCHITECTURE

In order to facilitate maintenance and scalability, the project has been designed to follow a modular approach. At the top level, the work has been divided into a set of modules. Each module in turn has parts and sub-parts. Working in modules has the advantage that the complete work can be done as an assemblage of small parts, each of which is easy to handle, debug, maintain and upgrade.

The top-level division of the project yields the following modules:

(a) Pre-processor: This does the basic job of cleaning up the incoming data of unwanted entities. It also does some initial formatting of the data, which makes the task of the later modules more regulated. The output from the preprocessor goes to the Lexical analyzer.

(b) Lexical Analyzer: Here the pre-processed data undergoes a few changes. Firstly, a set of values, which have been predecided, are marked with tags. In the next stage these tags along with the values they envelop are removed. This is because for lexical analysis there is a need to look for patterns in the data, and the presence of individual values tends to spoil the patterns. After this the whole data is produced in a tokenized format, now ready to be analyzed so that the formulation of the lexicon can be done.

(c) Syntactic Analyzer: The syntactic analyzer forms the core of the whole system. This is where the output from the previous stage is studied to produce the grammar, which is to be used for parsing the input text. The grammar has to be formulated so that it reads all the pertinent portions of the input even if they occur in varied forms.

(d) Constraint Extractor: This is the phase which produces the final output from the system. Its job is to read the output produced when the grammar parse the text, and write out the constraints accordingly in a proper format.

In the following, the details for the Pre-processing, Tagging and Removal of Tags are grouped together for the development and implementation steps with the differences pointed out wherever applicable.

3.1 PRE-PROCESSOR

The job of the pre-processor is to clean up and arrange the raw data which has been downloaded from the website www.travelocity.com in a format which is suitable to the following stages, eliminating unwanted parts of text, formatting the text in a format suitable for the next stage and also (as in this case) tagging the relevant parts of the text with suitable entity tags.

The pre-processor has been organized into the following sub-parts:

(a) Initial processing of data;
(b) Tagging;
(c) Removal of the tags;
which are now elucidated further:
The Architecture of the System
3.1.1 INITIAL PROCESSING OF INPUT DATA

The job of this part is to clean up the input data. After careful analysis of the structure of the input data, it was decided that a part of it is not useful for the current purposes. So it is up to the pre-processor to remove these parts of text from the input. This includes unwanted rules and any headings that those rules may be having. Removal of unimportant parts of the data is useful because there is little point in making that data go through the various stages of the project. This not only makes things simpler, but also means that the scripts and programs take lesser time to run.

This step is important because neither the lexicon nor the grammar should be influenced by text which holds no importance.

Another job accomplished by the pre-processor is the formatting of the data in a format for better handling by the modules to follow. In this part of the pre-processor, additional blank lines are added before and after every heading. Also, all the headings are put inside tags. This helps at later stages in the identification of the headings for various stages of parsing. It is important to tag the headings because they act as separators and are used at all stages as markers within the data. Thus the headings given for each rule play a pivotal role, by helping in the identification of boundaries in the data.

The pre-processor is implemented in Perl and Flex.

3.1.2 TAGGER

The purpose of this module is to mark certain parts of the data with special tags for later processing. This is required because while proceeding from the input data towards the goal of formalizing the lexicon, the variability and non-uniformity of text should be minimized. For this, it is essential that things like money, days, date, durations, country names, airline names, cities, etc… are removed from the data since we need not include them in the lexicon. So all the occurrences of the above mentioned types of data are substituted by the value enclosed between SGML-like tags representing the category of that value.

Example: Any occurrence of USD 100 (denoting 100 US Dollars) is replaced by <currency>USD 100</currency>

The list of all entities that are tagged is provided in the Appendix. For this module, vast lists containing names and corresponding codes for Airlines, Countries, Currencies and Cities have been used. These lists act as identifiers of the occurrence of the value which needs to be tagged.

Tagging ensures that all such values are easily identifiable and removable from the data as and when the need arises. This process is crucial because the lexicon and grammar will be built based on the text that this stage produces. Tagging ensures that the lexicon will be built on the basis of the occurrence of a category and not the actual values. Thus the tags and not the values will be a part of the lexicon so that they are parsed successfully.

The tagging phase is common both to the development and implementation phases.

3.1.3 TAG REMOVER

The tag remover is also a part of both the development and exploitation phases.

The various tags for different sets of values in the data have been inserted in the previous stage, now comes the time when they are to be removed so that values do not introduce highly varying
features in the lexicon. In actual fact, these removed values are not being discarded. They are to be
kept in a value list from which they will be extracted when the values are to be plugged into the
constraints during the exploitation part of the project.
This part of the project serves two purposes. Firstly, it is a part of the lexical analysis where it helps
in deciding about the contents of the lexicon. Secondly, it is also used to create a stack of values
which will be used later to furnish the final constraints.
The tag remover has been implemented in Perl.

3.2 LEXICAL ANALYSER

The objective of the lexical module is to identify words and phrases in the input set which recur
many times in the data so as to warrant an entry in the lexicon. This involves searching and looking
for text patterns in the data.
This stage is a part of the development process of the system. The lexicon is like the dictionary of
the system. It can understand only those words which are a part of its lexicon.
Not just words, but also groups of words can form a part of the lexicon. If a set of say, 5 words, is
in the lexicon, then it is read as just one single word. To find which groups of words should be in
the lexicon, the input text is analyzed to find the number of times every possible set of consecutive
words occur in the text. The most frequently occurring n-grams (n = 1 … 7) are kept in the lexicon.
In this project, groups ranging from length 1 to 7 were analyzed to look for prospective candidates
for the lexicon.
A few examples of n-grams which are in the lexicon are:
Depart from an intermediate point
Dptr from the farthest geographical point
Measured from dptr from gateway
Measured from dptr from origin to dptr from last stopover point

Lexical analysis comprises of the following parts:
(a) Tokenizer;
(b) Counters;

3.2.1 TOKENIZER

Tokenization is also a part of the development phase. It contributes towards building the lexicon.
Tokenization facilitates the counting of the number of occurrences of words and phrases. It is done
by separating the data set into tokens (a token is a set of homogeneous characters separated by a
separator). The separators have to be decided by analyzing the input data. The characters in Table 1
are treated as separators for the purpose of this project. This means that if a separator occurs
anywhere, the current token is supposed to be complete and a new token is expected to start after
the separator. The process of tokenization is needed because the entries in the lexicon are words
and thus they need to be separated. This process has another utility since it transforms the input into
a form which is ideal for the next stage which involves counting the frequency of occurrence of
words.
The tokenizer has been implemented as a Perl script.
Table 1 shows the separators used:

<table>
<thead>
<tr>
<th>CHARACTER</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;\n&quot;</td>
<td>New Line</td>
</tr>
<tr>
<td>&quot; &quot;</td>
<td>Space</td>
</tr>
<tr>
<td>&quot;.&quot;</td>
<td>Comma</td>
</tr>
<tr>
<td>&quot;:&quot;</td>
<td>Colon</td>
</tr>
<tr>
<td>&quot;,&quot;</td>
<td>Period</td>
</tr>
<tr>
<td>&quot;:-&quot;</td>
<td>Hyphen</td>
</tr>
<tr>
<td>&quot;:*&quot;</td>
<td>Asterisk</td>
</tr>
<tr>
<td>&quot;:/&quot;</td>
<td>Forward Slash</td>
</tr>
<tr>
<td>&quot;(&quot;</td>
<td>Right Bracket</td>
</tr>
<tr>
<td>&quot;)&quot;</td>
<td>Left Bracket</td>
</tr>
</tbody>
</table>

Table 1

3.2.2 COUNTERS
The final step before preparing the lexicon is counting the frequencies of n-grams of the tokens. This is also a part of the development phase.

This is done by using the tokenized output from the previous stage. The purpose of this is to find out those parts of data which occur more often than other parts. For this, the number of times every possible n-gram occurs is counted. Then on the basis of this frequency, it is decided whether an n-gram is present enough number of times to justify its inclusion in the lexicon.

For the current purposes it is more than sufficient to carry out this process for groups of words ranging in length from 1 upto 7 words. The output is generated as a list of all the n-grams in the data together with the frequency of each. By a comparative study of the output thus generated, it is decided as to which phrases are to be made a part of the lexicon.

The pertinent phrases as deduced from the n-grams are included in the lexicon. Once the lexicon has been formulated in this manner it is checked against the input to make sure that there are no tokens in the input which are not a part of the lexicon. Once this has been ensured, this stage of the project is complete.

The lexicon contains 702 lexical items plus the lists of Cities and Airline companies.

This generation of n-grams has been implemented in Perl.

3.3 SYNTACTIC ANALYSIS
Syntactic analysis is required both for the development and exploitation phases. In the development phase, it contributes towards writing the grammar, while during exploitation this step is used to parse the text with the grammar so as to help constraint extraction.

3.3.1 SENTENCE ALIGNER
The data is now passed through a stage called the sentence aligner. The anagram tool, which is being used for parsing, has the characteristic that it parses correctly only if it is fed one sentence per line. So the data is treated to remove any new line characters that occur in the middle of a sentence.

From analysis of the text, it was found that a sentences is terminated either by the use of a dot or the special characters (e.g. "\*\*\*\*"). Thus any new lines occurring before dot or "\*\*\*\*" in a sentence are
removed. This approach is stable within the considered context because the input to this part is devoid of any numbers, dates, other numerical data or abbreviations, since all of those have already been removed by the tagging and tag-removing steps. None of the previous modules introduces dots into the text. Therefore, a dot in the input can only be apart of an English sentence and thus it should act as a punctuation mark. So if a dot is encountered, one can be fairly certain that it actually denotes the end of the sentence.

The sentence aligner has been implemented in Perl.

3.3.2 DEVELOPMENT PHASE

The grammar has to be written manually. To carry out this part of the process, the development set is referred to.

This part of the work is like a bottom up approach. Before the grammar can be written, it is to be decided as to which parts of the input data are important enough to be parsed by the grammar. To do this, the input data is analyzed in detail keeping in mind the fact that constraints have to be extracted out of it. Those parts of the data that furnish important facts, are the ones that are of relevance. Once the important parts of the data are identified, the grammar is formulated to meet the requirements of parsing them.

For every condition, all possible phrases that occur in the data are extracted. Keeping in mind all the sentences that need to be parsed, non-terminals are defined. These non-terminals act as symbols that stand for a group of tokens of the text. Non-terminals are then put together to form production rules that identify with whole sentences of the text. This process is repeated for all the important phrases. These phrases should then be recognized by the grammar. An occurrence of such a phrase would mean that the corresponding condition or part of the condition has been detected and proper action needs to be taken. This process is repeated for all the conditions that had been found, and on completion of this, the grammar is obtained.

The grammar consists of 174 production rules.

For the development phase of the project, a set of 164 texts (containing 114204 words) was downloaded and used.

It is important that the grammar parses a good amount of the input data i.e. it is sufficiently general so that there is no risk of missing out on some pertinent parts.

3.3.3 EXPLOITATION PHASE

After the grammar has been formulated, the automatic syntactic analysis of data can be performed.

Here, the data is parsed with the grammar. For this the tool "anagram" provided by the SLPToolkit is used. The "anagram" binary takes the lexicon and the grammar in special formats and attempts to parse any input data with the grammar. It provides various modes of stating the output e.g. tree format and parentheses format. For the current purposes, it suffices to have the output in the parentheses format.

To make the system more robust, two separate sets of data were used. One is the development set, based on the analysis of which, the framework of the whole system is designed. The other set of data, which is independent of the development set is the validation set. This validation set acts as a test bed for looking for possible loopholes in the system and for further enhancements to make sure that the system can handle a wide variety of input. The validation set consists of 26 texts containing 19567 words.
3.4 CONSTRAINT EXTRACTOR

Up to this stage, the input data has been pre-processed and parsed. Now comes the final and most important stage of building the constraints. From the output of the parser, the job is to find out which constraint has been detected. This is done by finding out which sentence has been parsed by looking at the output from anagram. The output is searched for the occurrence of sequences of some specific non-terminals of the grammar, which would decide beyond doubt, which sentence has been parsed. This pattern matching needs to be very accurate because a stray occurrence of a match can trigger the output of a false constraint. Thus the matches are based on strict conditions.

After finding out that a particular sentence has occurred the work that remains to be done is to output the corresponding constraint with the values that should occur with it. For this, the stack of values that was created in the pre-processing stage is needed. Depending upon the constraint, the appropriate number of values is popped from the value stack and these popped values are printed as required. There may also be some values in the stack which may not be a part of some constraint. These are the values which are tagged by the entity tagger, but are not a part of a constraint forming sentence. These have to be detected by looking for the occurrence of an entity tag in a sentence which is known not to be a part of any constraint. When this happens, the stack is popped once so that the value is removed.

The constraint extractor has been implemented in Perl.

The constraints are given as output in a format which should be useful when using these in a Constraint Satisfaction Problem.

The output makes use of the above variables to denote concepts. These are the concepts that are the most pertinent in the set of airline rules. Although a model for the CSP for such a system is not available yet, the concepts chosen are generic and should fit any model which is developed.

All of the above steps contribute to produce constraints from the original text data. It is important that no details are left out or ignored at this stage, because any information that is lost here is not retrievable at a later stage. This has been ensured by working in parallel with the whole natural language full text. At all stages, the system works by looking both at the input from the previous stage and also the text. A good example of this is in the constraint extractor itself where values from the value stack are either used or discarded depending upon whether they occur in a sentence which has been parsed correctly or not. Thus care has been taken to incorporate all pertinent points within the radius of this endeavor.

A sample of input data and the corresponding set of constraints generated is provided in the following pages for the reader’s perusal.
3.5 AN EXAMPLE

An example is provided on the following pages to illustrate the functioning of the system. The raw text is written on the even page and aligned to it on the odd page is the pre-processed text.

3.5.1 The constraints generated:

The following are the constraints that are generated from the input text given on the following pages.

Flight = "AOM"
From = “Geneva, Switzerland (GVA)”
TO = “Sydney, Australia (SYD)”
Flight Number = “BLSX6MSW”
Booking Code = “B – “
IF { Tcancel < To } THEN { CHARGE(CANCEL: CANCEL,NO-SHOW,REFUND) = CHF200*(1 - CHILD_DISCOUNT) }
IF { DEATH OF PASSENGER OR FAMILY MEMBER } THEN { TKT_REFUND = TRUE AND CHARGE = ZERO }
IF { PASSENGER = INFANT AND PRICE = 0.1*ADT_FARE } THEN { CHARGE = ZERO }
IF {} THEN { CHARGE(CHANGE: REISSUE,REVALIDATION) = CHF200*(1 - CHILD_DISCOUNT) }
IF { PASSENGER = INFANT AND PRICE = 0.1*ADT_FARE } THEN { CHARGE = ZERO }
Tkt = Tres
T1 >= To + 7DAYS
MINSTAY = 7DAYS
T1 <= To + 6MONTHS
MAXSTAY = 6MONTHS
FREE_STOP(CMB,PAR) = ONE
PAID_STOP(CMB,PAR) = ONE ADT_FARE = CHF150.00 CHD_FARE = CHF100.50
INF_FARE = CHF15.00
STOPOVER ===> STAY >= 24HOURS
TRANSFERS = 4TRANSFERS
TRN_DIR(PAR) = 1
TRN_DIR(CMB) = 1
3.5.2 The original file taken from the web (page 1):

Fare Rule Details

Flight: AOM
From: Geneva, Switzerland (GVA)
To: Sydney, Australia (SYD)

LEGEND for abbreviations
Fare details for BLX6MSW
SUPER SPECIAL INSTANT PURCH SPECIAL FARE FARE
BK CODE
B-

PENALTY
CANCELLATIONS - BEFORE DEPARTURE CHARGE CHF 200 FOR
CANCEL/ NO-SHOW/ REFUND. CHILD/ INFANT DISCOUNTS - APPLY.
WAIVED IN THE CASE OF DEATH OF PASSENGER/ FAMILY MEMBER.
PENALTY CHARGE DOES NOT APPLY TO INFANTS PAYING 10 PCT OF
THE ADULT FARE. CANCELLATIONS - AFTER DEPARTURE - TICKET
IS NONREFUNDABLE. CHANGES - ANY TIME CHARGE CHF 200 FOR
REISSUE/ REVALIDATION. CHILD/ INFANT DISCOUNTS - APPLY.
PENALTY CHARGE DOES NOT APPLY TO INFANTS PAYING 10 PCT OF
THE ADULT FARE.

RES/TKG
TKT MUST BE ISSUED SAME DAY RES MADE. SGMTS USING THIS
RULE MUST BE CONFIRMED.

MIN STAY
7 DAYS. Measured from DPTR FROM GATEWAY.

MAX STAY
RETURN TRVL MUST COMMENCE NO LATER THAN 6 MONTHS. Measured
FROM DPTR FROM ORIGIN TO DPTR FROM LAST STOPOVER POINT.

SEASONS
1OCT-15NOV00, 16-31JAN01 OR FEB01. SEASON IS BASED ON DATE
OF ORIGIN AND IS DETERMINED BY DATE OF DEPARTURE OF FIRST
INTERNATIONAL SECTOR.

BLACKOUTS
NO BLACKOUT RESTRICTIONS APPLY.

FLT APPL
NOTE: FOR TRAVEL COMMENCING ON/AFTER 01APR00 AS PER
SPECIFIED ROUTING ONLY RBD BETWEEN ZRH/GVA AND PAR ON IW -
B BETWEEN PAR AND SYD ON IW - B

STOPOVERS
A FREE STOPOVER IS PERMITTED AT CMB/PAR, IF IN ROUTING
APPLICABLE TO FARE AND AN ADDITIONAL STOPOVER IS PERMITTED
AT CMB/PAR, AT CHF 150.00 EACH PER ADT AND CHF 100.50 EACH
PER CHD AND CHF 15.00 EACH PER INF, IF IN ROUTING
APPLICABLE TO FARE. A STOPOVER OCCURS WHEN THE PASSENGER
DOES NOT DEPART AN INTERMEDIATE POINT WITHIN 24 HOURS.
3.5.3 The file after pre-processing (page 1):

<from>Geneva, Switzerland (GVA)
<to>Sydney, Australia (SYD)

<flight number> BLS<number>6</number>MSW
<BK CODE>
B -

<PENALTY>
CANCELLATIONS - BEFORE DEPARTURE CHARGE <price>CHF 200</price> FOR CANCEL/ NO-SHOW/ REFUND. CHILD/ INFANT DISCOUNTS - APPLY. WAIVED IN THE CASE OF DEATH OF PASSENGER/ FAMILY MEMBER. PENALTY CHARGE DOES NOT APPLY TO INFANTS PAYING <percent>10 PCT</percent> OF THE ADULT FARE. CANCELLATIONS - AFTER DEPARTURE - TICKET IS NONREFUNDABLE. CHANGES - ANY TIME CHARGE <price>CHF 200</price> FOR REISSUE/ REVALIDATION. CHILD/ INFANT DISCOUNTS - APPLY. PENALTY CHARGE DOES NOT APPLY TO INFANTS PAYING <percent>10 PCT</percent> OF THE ADULT FARE.

<RES/TKG>
TKT MUST BE ISSUED SAME DAY RES MADE. SGMTS USING THIS RULE MUST BE CONFIRMED.

<MIN STAY>
<days>7 DAYS</days>. MEASURED FROM DPTR FROM GATEWAY.

<MAX STAY>
RETURN TRVL MUST COMMENCE NO LATER THAN <months>6 MONTHS</months>. MEASURED FROM DPTR FROM ORIGIN TO DPTR FROM LAST STOPOVER POINT.

<FLT APPL>
NOTE: FOR TRAVEL COMMENCING ON/AFTER <date>01 APR00</date> AS PER SPECIFIED ROUTING ONLY RBD BETWEEN ZRH/GVA AND PAR ON IW - B BETWEEN PAR AND SYD ON IW - B

<STOPOVERS>
A FREE STOPOVER IS PERMITTED AT CMB/Par, IF IN ROUTING APPLICABLE TO FARE AND AN ADDITIONAL STOPOVER IS PERMITTED AT CMB/Par, AT <price>CHF 150.00</price> EACH PER ADT AND <price>CHF 100.50</price> EACH PER CHD AND <price>CHF 15.00</price> EACH PER INF, IF IN ROUTING APPLICABLE TO FARE. A STOPOVER OCCURS WHEN THE PASSENGER DOES NOT DEPART AN INTERMEDIATE POINT WITHIN <hours>24 HOURS</hours>. 


3.5.4 The original file taken from the web (page 2):

TICKET RESTRICTIONS
NO SALES RESTRICTION APPLY.

DISCOUNTS
67 PCT OF THE ADT FARE IS CHARGED FOR A CHD PSGR 2 THRU 11 YRS OF AGE. CHD PSGR MUST BE ACCOMPANIED ON ALL SGMTS BY AN ADT PSGR TRAVELING IN THE SAME COMPARTMENT AT A FARE GOVERNEED BY THE SAME RULE. -- -- -- 10 PCT OF THE ADT FARE IS CHARGED FOR AN INF PSGR UNDER 2 YRS OF AGE NOT OCCUPYING A SEAT. INF PSGR MUST BE ACCOMPANIED ON ALL SGMTS BY AN ADT PSGR TRAVELING IN THE SAME COMPARTMENT AT A FARE GOVERNEED BY THE SAME RULE. *** UNACCOMPANIED CHILD 5 THRU 11 YEARS PERMITTED TO TRAVEL AT THE CHILD'S DISCOUNT. INFANTS OCCUPYING A SEAT WILL BE PERMITTED AT THE CHILD's DISCOUNT. NOTE - MUST BE ACCOMPANIED ON ALL FLIGHTS IN SAME COMPARTMENT BY ADULT 15 OR OLDER. TOUR CONDUCTOR/SALES AGENT DISCOUNT DOES NOT APPLY.

REROUTE
SEE PENALTY.

TRANSFERS
4 TRANSFERS PERMITTED. --1 IN EACH DIRECTION IN PAR. --1 IN EACH DIRECTION IN CMB.

COMBINATIONS
CARRIER / RULE / FARE CIRCLE OPEN ROUND BASIS TRIP JAW TRIP SAME / SAME / SAME C C C SAME / SAME / DIFF C C C SAME / DIFF / SAME C C C SAME / DIFF / DIFF C C C DIFF / SAME / SAME C C C DIFF / SAME / DIFF C C C DIFF / DIFF / SAME C C C DIFF / DIFF / DIFF C C C / PERMITTED N / NOT PERMITTED -- / DOES NOT APPLY C / SEE TEXT FOR CONDITIONS CONDITIONS APPLICABLE TO ROUND TRIP/CIRCLE TRIP/SINGLE OR DOUBLE OPEN JAW -- MAY BE COMBINED WITH ANY ONE WAY/ROUND TRIP FARE OF THE SAME FARE TYPE GOVERNEED BY ANY RULE IN IPREUAS PROVIDED TRAVEL BETWEEN SWITZERLAND AND AUSTRALIA IS VIA IN/SR. -- MOST RESTRICTIVE CONDITIONS APPLY. ADDITIONAL CONDITIONS APPLY ROUND TRIP -- PERMITTED. CIRCLE TRIP -- PERMITTED. OPEN JAW -- SINGLE OPEN JAW PERMITTED AT EITHER ORIGIN OR DESTINATION. DOUBLE OPEN JAW PERMITTED. END-ON-END -- NOT PERMITTED.

OPEN RTN
NOT ALLOWED.

REFUNDS
SEE PENALTY.

CO-TERMINALS
NOT APPLICABLE.

MISC
NOTE - TICKETS MUST SHOW BY THE USE OF AN INSERT OR STICKER THAT TRAVEL IS AT A SPECIAL FARE AND SUBJECT TO SPECIAL CONDITIONS.
3.5.5 The file after pre-processing (page 2):

<REROUTE>
SEE PENALTY.

<TRANSFERS>

<trns>4 TRANSFERS</trns> PERMITTED. -<number>1</number> IN EACH DIRECTION IN PAR. -<number>1</number> IN EACH DIRECTION IN CMB.

<OPEN RTN>
NOT ALLOWED.

<REFUNDS>
SEE PENALTY.

<MISC>

NOTE - TICKETS MUST SHOW BY THE USE OF AN INSERT OR STICKER THAT TRAVEL IS AT A SPECIAL FARE AND SUBJECT TO SPECIAL CONDITIONS.
4 CONCLUSION

With all of the aforementioned work completed, the output from the system can be sent to the Constraint Satisfaction Problem where the constraints will be resolved. The most engrossing part of the work was the fact that because of the special nature of the input text, the standard lexicon for English and related tools could not be used. With the system having its own tailor-made lexicon, it is very efficient and accurate. Although no efforts have been spared to ensure that the system does not miss out on probable constraints, but with a much larger data set there could be scope for introducing more robustness into the system. The performance of the system on the data on which it has been tested is extremely good. However, a real evaluation requires an actual test set with annotated data, which means that an expertise which we do not currently have is needed to have a full evaluation.

5 FUTURE WORK

Some parts of the input text were a priori decided to be left out of the system due to lack of information about their meaning and significance. As future work, it is suggested that the text under the heading “Discounts” also be included for analysis and constraints produced from it. Also, the text referring to “Upgrade Penalty” that is, the penalty incurred on changing an upgraded ticket, could be analyzed for its usefulness as a constraint and included as a part of the system if need be.
6 APPENDIX

6.1 Variables in the Constraints:

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight</td>
<td>Name of the airline company.</td>
</tr>
<tr>
<td>From</td>
<td>Originating point of the journey.</td>
</tr>
<tr>
<td>TO</td>
<td>Terminating point of the journey.</td>
</tr>
<tr>
<td>BookingCode</td>
<td>The booking code.</td>
</tr>
<tr>
<td>FlightNumber</td>
<td>The flight number.</td>
</tr>
<tr>
<td>Tkt</td>
<td>Indicates the day when the ticket is bought.</td>
</tr>
<tr>
<td>Tres</td>
<td>Indicates the day when reservation is made.</td>
</tr>
<tr>
<td>To</td>
<td>The day of departure from the originating point of the journey.</td>
</tr>
<tr>
<td>Tl</td>
<td>The day of departure from the last stopover point.</td>
</tr>
<tr>
<td>Tcancel</td>
<td>The day when the ticket is cancelled.</td>
</tr>
<tr>
<td>Tchange</td>
<td>The day when the itinerary is changed.</td>
</tr>
<tr>
<td>Charge(Change)</td>
<td>Indicates the money charged when the ticket is cancelled.</td>
</tr>
<tr>
<td>Charge(Cancel)</td>
<td>Indicates the money charged when the itinerary is changed.</td>
</tr>
<tr>
<td>Stopovers</td>
<td>The number of stopovers.</td>
</tr>
<tr>
<td>Transfers</td>
<td>The number of transfers allowed.</td>
</tr>
<tr>
<td>Tkt_Refund</td>
<td>Indicates whether ticket will be refunded or not.</td>
</tr>
<tr>
<td>Rbk_Reroute</td>
<td>Indicates whether rebooking and rerouting are permitted or not.</td>
</tr>
<tr>
<td>Free_Stop(places)</td>
<td>Stands for the number of free stopovers allowed in places.</td>
</tr>
<tr>
<td>Paid_Stop(places)</td>
<td>Stands for the number of paid stopovers allowed in places.</td>
</tr>
<tr>
<td>Trn_Dir(places)</td>
<td>Stands for the number of transfers allowed in each direction at cities.</td>
</tr>
<tr>
<td>Passenger</td>
<td>Indicates the age group of the passenger.</td>
</tr>
<tr>
<td>ADT_FARE</td>
<td>The fare charged for an adult.</td>
</tr>
<tr>
<td>CHD_FARE</td>
<td>The fare charged for a child.</td>
</tr>
<tr>
<td>INF_FARE</td>
<td>The fare charged for an infant.</td>
</tr>
<tr>
<td>MINSTAY</td>
<td>The minimum duration between the departure from origin and from the last stopover point.</td>
</tr>
<tr>
<td>MAXSTAY</td>
<td>The maximum duration between the departure from origin and from the last stopover point.</td>
</tr>
</tbody>
</table>

Table 1: The list of variables that are used to report the constraints, and their meanings.
6.2 The Tag Set

<table>
<thead>
<tr>
<th>TAG</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price</td>
<td>Indicates the price in any currency e.g. USD 100.</td>
</tr>
<tr>
<td>Hour</td>
<td>Indicates time e.g. 5:00 p.m..</td>
</tr>
<tr>
<td>Country</td>
<td>This tag is used for tagging country names.</td>
</tr>
<tr>
<td>Duration</td>
<td>Used to tag any time duration of the form Date-Date or Weekday-Weekday.</td>
</tr>
<tr>
<td>Weekday</td>
<td>This is the tag for all weekdays from Monday to Sunday</td>
</tr>
<tr>
<td>Date</td>
<td>This is the tag for all occurrences of dates in the text.</td>
</tr>
<tr>
<td>nth</td>
<td>This tags occurrences like 1st, 5th etc.</td>
</tr>
<tr>
<td>Percent</td>
<td>This is the tag for occurrences of percentage values e.g. 10 PCT</td>
</tr>
<tr>
<td>Months</td>
<td>This is used for tagging durations that occur in terms of months.</td>
</tr>
<tr>
<td>Days</td>
<td>This is used for tagging durations that occur in terms of days</td>
</tr>
<tr>
<td>Stopovers</td>
<td>This tags the number of stopovers e.g. 2 Stopovers.</td>
</tr>
<tr>
<td>Transfers</td>
<td>This tags occurrences referring to number of transfers.</td>
</tr>
<tr>
<td>Period</td>
<td>This tags time durations that occur in the form of hour-hour e.g. 4:00 pm – 5:00 pm.</td>
</tr>
<tr>
<td>Provision</td>
<td>This is the tag for references to provision numbers.</td>
</tr>
<tr>
<td>Rule</td>
<td>This is the tag for references to rules.</td>
</tr>
<tr>
<td>Dl</td>
<td>This is used to tag the multiple occurrence of dashes together.</td>
</tr>
<tr>
<td>Age</td>
<td>This is the tag for references made to the age of passengers.</td>
</tr>
<tr>
<td>Agefrom</td>
<td>This is the tag for text referring to age between certain years e.g. 8 thru 10 years.</td>
</tr>
<tr>
<td>Number</td>
<td>This is the tag for occurrences of numbers occurring at places which do not come under the above criteria.</td>
</tr>
</tbody>
</table>

Table2: The Tag Set
6.3 Grammar
The grammar consists of various non-terminals. The top-level symbol for the grammar is the non-terminal \( S \). Then the next level in the hierarchy is that of the clauses which have the symbol \( C \). A clause can be composed of many different non-terminals. A few non-terminals denoting relational operators have been introduced. For example “\( \text{LEQCBF} \)” stands for “less than or equal to”, “\( \text{GEQC} \)” stands for “greater than or equal to”, “\( \text{EQC} \)” stands for “equal to”, etc.

Here comes the grammar:

\[
\begin{align*}
S & \rightarrow \ C \ OR_\ C \ CONJ \\
S & \rightarrow \ C \\
C & \rightarrow \ \text{RES}_\ TO \\
C & \rightarrow \ \text{TKTPUR} \\
C & \rightarrow \ \text{RET} \\
C & \rightarrow \ \text{STOP} \\
C & \rightarrow \ \text{NUMSTOP} \\
C & \rightarrow \ \text{TRANS} \\
C & \rightarrow \ \text{TRANS}_\ CT \\
C & \rightarrow \ \text{TKT}\_\ RES \\
C & \rightarrow \ \text{REFUND}_\ \\
C & \rightarrow \ \text{WAIV} \\
C & \rightarrow \ \text{REBOOK} \\
C & \rightarrow \ \text{PEN} \\
C & \rightarrow \ \text{REOU} \\
C & \rightarrow \ \text{STP}\_\ PRICE \\
C & \rightarrow \ \text{TRNF}\_\ DIR \\
C & \rightarrow \ \text{STP}\_\ AT \\
C & \rightarrow \ \text{CANC}\_\ DEPTIME \\
C & \rightarrow \ \text{CANC}\_\ CHG \\
C & \rightarrow \ \text{CHANGE}_\ \\
C & \rightarrow \ \text{DISC} \\
C & \rightarrow \ \text{NUMSTOP}_\ FR \\
\text{RES}_\ TO & \rightarrow \ \text{TIME}_\ \text{MUST1}_\ \text{LEQCBF}_\ \text{PUNCT} \\
\text{TKTPUR} & \rightarrow \ \text{S1}_\ OR\_\ S2\ \text{CONJ}_\ \text{PUNCT} \\
\text{TKTPUR} & \rightarrow \ \text{TIME}_\ \text{MUST1}_\ \text{EQC}_\ \text{PUNCT} \\
\text{RET} & \rightarrow \ \text{TIME}_\ \text{GEQC}_\ \text{SPEC}\_\ \text{DETAIL}_\ \text{PUNCT} \\
\text{RET} & \rightarrow \ \text{MEASURE}_\ \text{DETAIL}_\ \text{PUNCT} \\
\text{RET} & \rightarrow \ \text{TIME}_\ \text{GEQC}_\ \text{SPEC}_\ \text{TIME}_\ \text{PUNCT} \\
\text{RET} & \rightarrow \ \text{TIME}_\ \text{MUST1}_\ \text{LEQCBF}_\ \text{DETAIL}_\ \text{PUNCT} \\
\text{STOP} & \rightarrow \ \text{STOPOVER}_\ \text{OCCURS}_\ \text{NOT}_\ \text{DEPART}_\ \text{LEQCBF}_\ \text{PUNCT} \\
\text{NUMSTOP} & \rightarrow \ \text{INF}_\ \text{NUM}_\ \text{STP}_\ \text{PERMITTED}_\ \text{PUNCT} \\
\text{NUMSTOP} & \rightarrow \ \text{ZERO}_\ \text{NUM}_\ \text{STP}_\ \text{PERMITTED}_\ \text{PUNCT} \\
\text{NUMSTOP} & \rightarrow \ \text{ONE}_\ \text{NUM}_\ \text{STP}_\ \text{PERMITTED}_\ \text{PUNCT} \\
\text{NUMSTOP} & \rightarrow \ \text{STP}\_\ PLACES\ \text{PUNCT}\ \text{JUNK}\ \text{STP}_\ \text{PLACES}_\ \text{PUNCT}_\ \text{PREPO}_\ \text{PRICE}_\ \text{PER}_\ \text{PUNCT} \\
\text{NUMSTOP} & \rightarrow \ \text{NUMSTOP}_\ \text{INTERNAL}_\ \text{PUNCT} \\
\text{NUMSTOP}_\ FR & \rightarrow \ \text{STOPOVER}_\ \text{BE}_\ \text{PERMITTED}_\ \text{PUNCT}_\ \text{PREPO}_\ \text{PRICE}_\ \text{PER}_\ \text{PUNCT} \\
\text{TRANS} & \rightarrow \ \text{NUM}_\ \text{TRN}_\ \text{PERMITTED}_\ \text{ON}_\ \text{INTERNAL}_\ \text{PUNCT} \\
\text{TRANS} & \rightarrow \ \text{NUM}_\ \text{TRN}_\ \text{PERMITTED}_\ \text{PUNCT} \\
\text{TRANS} & \rightarrow \ \text{INF}_\ \text{TRANS} \\
\text{TRANS} & \rightarrow \ \text{NOT}_\ \text{TRANS} \\
\text{TRANS} & \rightarrow \ \text{NUM}_\ \text{TRN}_\ \text{PERMITTED}_\ \text{PUNCT}_\ \text{EACHDIR}_\ \text{PUNCT} \\
\text{TRANS} & \rightarrow \ \text{OR}_\ \text{NUM}_\ \text{TRN}_\ \text{PERMITTED}_\ \text{PUNCT}_\ \text{EACHDIR}_\ \text{PUNCT} \\
\end{align*}
\]
CANC_DEPTIME ---> ACTION BEF_T NOUN FEE PHRASE ACTION_ PUNCT
AGEGRP_ NOUN VERB PUNCT
CANC_DEPTIME ---> AFT COST BE_ NOT PUNCT
CANC_DEPTIME ---> ACTION_PERMITTED_ BEF_T PUNCT
CHANGE_ ---> NOUN BEF_T CHG_ PUNCT
CHANGE_ ---> NOUN PUNCTUATION AFT_T PUNCTUATION CHG_ 
PUNCT
CHANGE_ ---> PHRASE PUNCTUATION NOUN AFT_T CHG_ PUNCT
CHANGE_ ---> NOUN PUNCTUATION ANYTIME_ CHG_ PUNCT
CHANGE_ ---> PRE PHRASE PERMITTED_ PUNCT
CHANGE_ ---> PRE PUNCTUATION NOUN PHRASE PERMITTED_ 
PUNCT
CHANGE_ ---> NOUN ANYTIME_ CHG_ PUNCT
CHANGE_ ---> NOUN PERMITTED_ PREPO ANYTIME_ PUNCT
CHANGE_ ---> NOUN PUNCTUATION ANYTIME_ ARTICLE NOUN OF_ 
PRICE PENALTY_ VERB PHRASE NOUN PREP PHRASE 
PUNCT
CHANGE_ ---> PENALTYCHNG BEF_T BE_ PRICE PUNCTUATION 
INTERNAL PUNCT
CHANGE_ ---> PENALTYCHNG AFT_T BE_ PRICE PUNCTUATION 
INTERNAL PUNCT
DISC ---> AGEGRP_ NOUN VERB PUNCT
DISC ---> AGEGRP_ NOUN PUNCTUATION VERB PUNCT
LEQCBF ---> BE SUBLEQCBF
LEQCBF ---> SUBLEQCBF
SUBLEQCBF ---> LE MEASURE
SUBLEQCBF ---> LE MEASURE BEF TIME
GEQC ---> VALID ON
SPEC ---> NTH WEEKD AFT HOUR
SPEC ---> NTH WEEKD
MEASURE ---> DAYS
MEASURE ---> HOURS
MEASURE ---> MONTHS
BE ---> BE_ PPV
BE ---> BE2_
LEQCAF ---> BE LE SUBLEQCAF
LEQCAF ---> SUBLEQCAF
SUBLEQCAF ---> MEASURE AFT TIME BE
PERMITTED_ ---> BE_ PERMITTED_
EQC ---> BE EQ TIME BE
EQC ---> BE EQ TIME PPV
S1 ---> TIME MUST1 LEQCBF
S2 ---> LEQCAF
CONJ ---> PUNCT CLAUSE
DETAIL_ ---> PUNCT DETAIL
TIME ---> TTKT
TIME ---> TRES
TIME ---> TO
TIME ---> TL
IL_ ---> IL
IL_ ---> IL OR_ IL
PERSON_ ---> PERSON
PERSON_ ---> PERSON OR_ PERSON
SENT ---> IL_ OF_ PERSON_
SENT ---> SENT OR_ SENT
PLACE ---> CITY
6.4 Software Organization

The project has been organized as a set of four directories. The directory `src/` contains the files needed to build the system. The `bin/` directory contains all the scripts and executable binaries of the system. The `doc/` directory contains the soft copy of the report and the presentation slides. The directory `data/` contains the initial, pre-processed and output forms of the data for the development and exploitation phases. It also contains the grammar and the lexicon that were built.

To build the system, simply type

```
  gmake install
```

in `src/` directory.

To launch the system, first the file `PreProc.pl` in `bin/` should be executed as under:

```
  ./PreProc.pl input_text_file output_file
```

Then to extract constraints the file `Build_Constraints.pl` should be executed as under:

```
  ./Build_Constraints.pl input_file stack_file output_file
```

Here, the input file for `Build_Constraints.pl` should be the same as the output file from `PreProc.pl`. 