

Temper

Temporal Programmer: An Introduction

Abbas K. Zaidi
Alexander H. Levis
<szaidi2>, <alevis>@gmu.edu

Adversary Behavioral Modeling
Maxwell AFB, Montgomery AL
March 8 – 9, 2007

System Architectures
Laboratory
George Mason University



Temper

viking.gmu.edu



- **A Logic for Time**
 - **Point-Interval Logic**
 - **Point Graphs**
- **Temper** – **Software Implementation of Point-Interval Formalism**
- **Temporal Issues in Forensics**
- **Example: Applying Temper to London Bombing Data**



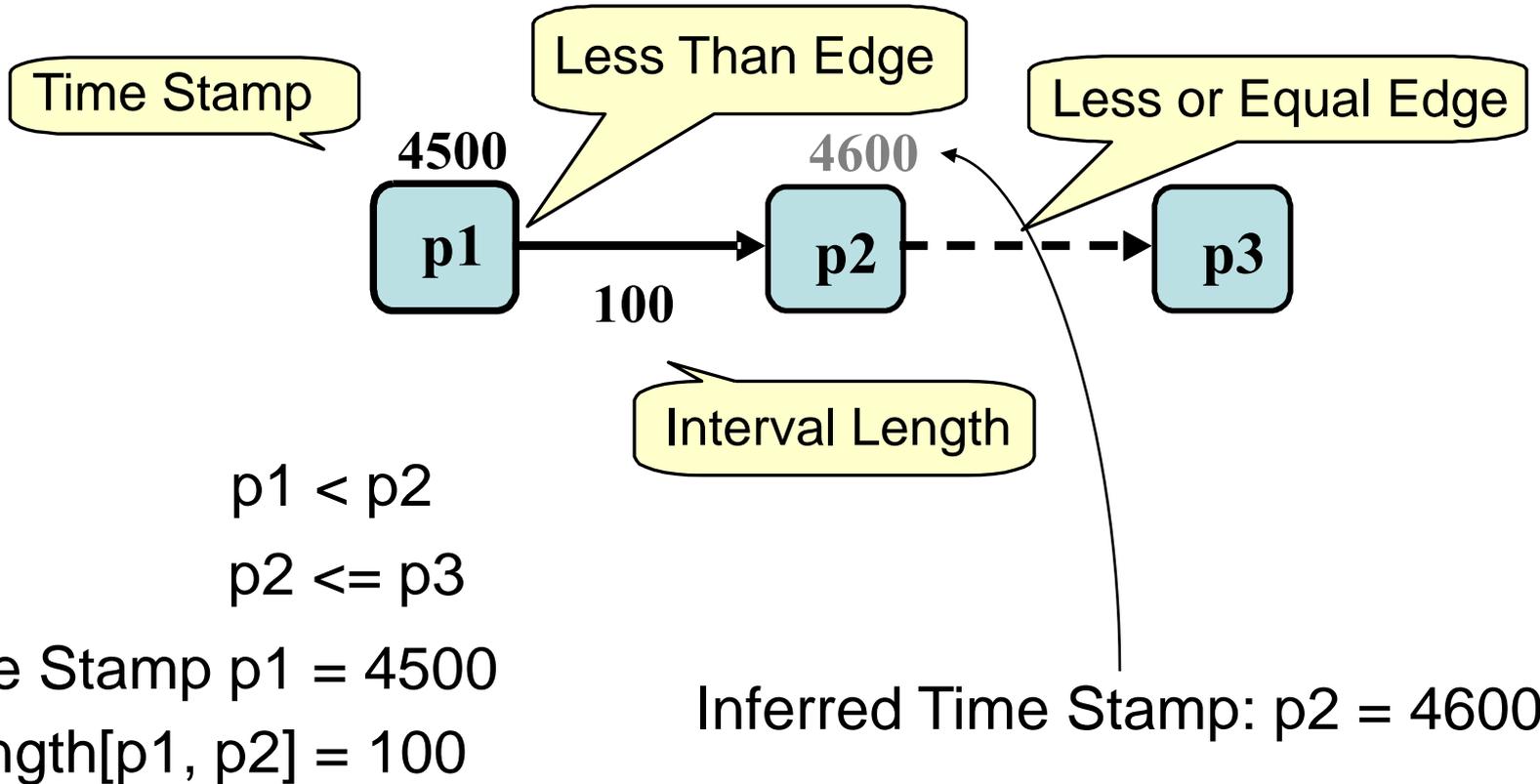
- Logic is the “Art of Reasoning”
- Logic is used to make inferences based on the available information
- Formal logic makes inferences based purely on the form of the content, without any understanding of the meaning of the content
- Reasoning based just on the form is important because this means computers can do it
- A formal logic for time enables us to:
 - Characterize time-sensitive attributes of a domain to be modeled
 - Do temporal analysis of a domain, which helps us in developing a better understanding of the relationship between domain entities
 - Identify Inconsistencies and anomalies

There are approaches that allow explicit representation of time and reasoning about it. The use of time logics to forensics is proposed.

- Allen introduced Interval Algebra as a framework for temporal reasoning. The algebra takes time intervals to be primitives. There are 13 possible relationships between a pair of intervals:

$R = \{\text{before, meets, overlaps, starts, during, finishes, equals, after, met-by, started-by, contains, finished-by}\}$

- We add points (intervals of zero length) and expand the set of relationships to define a Point Interval Logic (PIL)
- Knowledge Representation
 - A graph with nodes representing time points and edges representing the ‘inequalities’ captures the information in PIL statements



Point Interval Logic Statements and the corresponding Point Graph

```

sB < eB
sC < eC
sD < eD
sF < eF
sG < eG
sH < eH
Length [sB,eB] = 8
Length [sC,eC] = 9
Length [sD,eD] = 4
Length [sF,eF] = 1

```

Point Graph

Input Window

System Architectures Laboratory
George Mason University

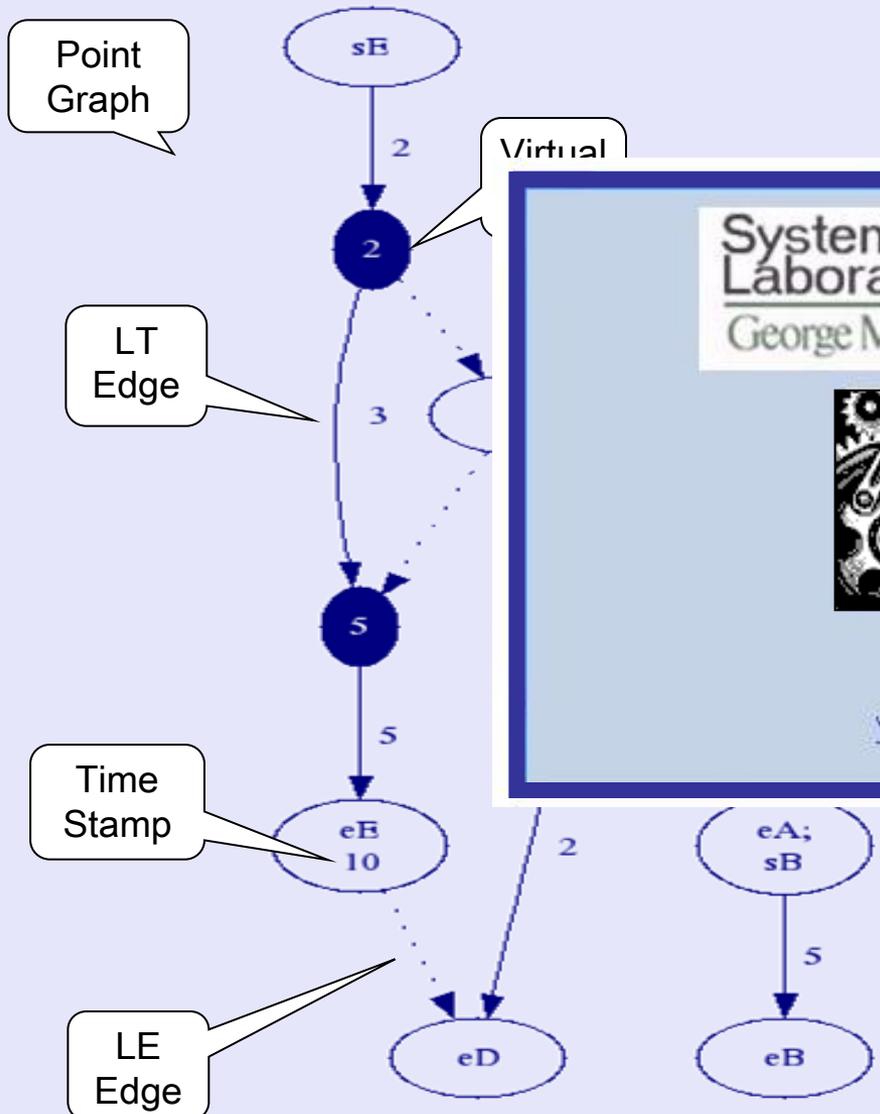
Temper
viking.gmu.edu

LT Edge

Time Stamp

LE Edge

Output Window



Earliest Start Time	Latest End Time
0	8
0	12
8	12
9	13
12	15
10	15

Temper – The Software



- **Temper** is a tool for temporal knowledge representation, reasoning, and planning using **Point-Interval Logic (PIL)**.
- **PIL** is a formal algebraic framework for reasoning with time. It has the ability to handle both:
 - Events and Activities
 - Quantitative and Qualitative temporal relationships
 - Reasoning and Planning
- The relationships among various activities and events in a domain are specified in the form of **PIL** statements. These statements are converted into a graphical construct called **Point Graphs (PG)**.
 - Algorithms for verification, inference, and planning are implemented on the Point Graph representation.
- The implementation of **PIL** is in the form of a .NET class library called **PIL Engine**. It provides an application programming interface (API) that can be used in any .NET compliant programming language. It uses **QuickGraph**, which is an open-source C# implementation of the **Graphviz** library from AT&T.
- **Temper** provides a graphical user interface (GUI) to **PIL Engine**.

Add/Delete PIL Statements

Add Stamp	Delete Stamp	[Dropdown]	=	[Dropdown]	[Text]
Add Length	Delete Length	[Dropdown]	[Dropdown]	=	[Dropdown]
Add Relation	Delete Relation	[Dropdown]	<	[Dropdown]	[Dropdown]
Add Composite Relation		[Dropdown]	<	[Dropdown]	[Dropdown]
Reference Date and Time		Wednesday, October 11, 2006		24:00	

Numeric Stamp

Day Hour Minute Second

Language Editor

Query Editor

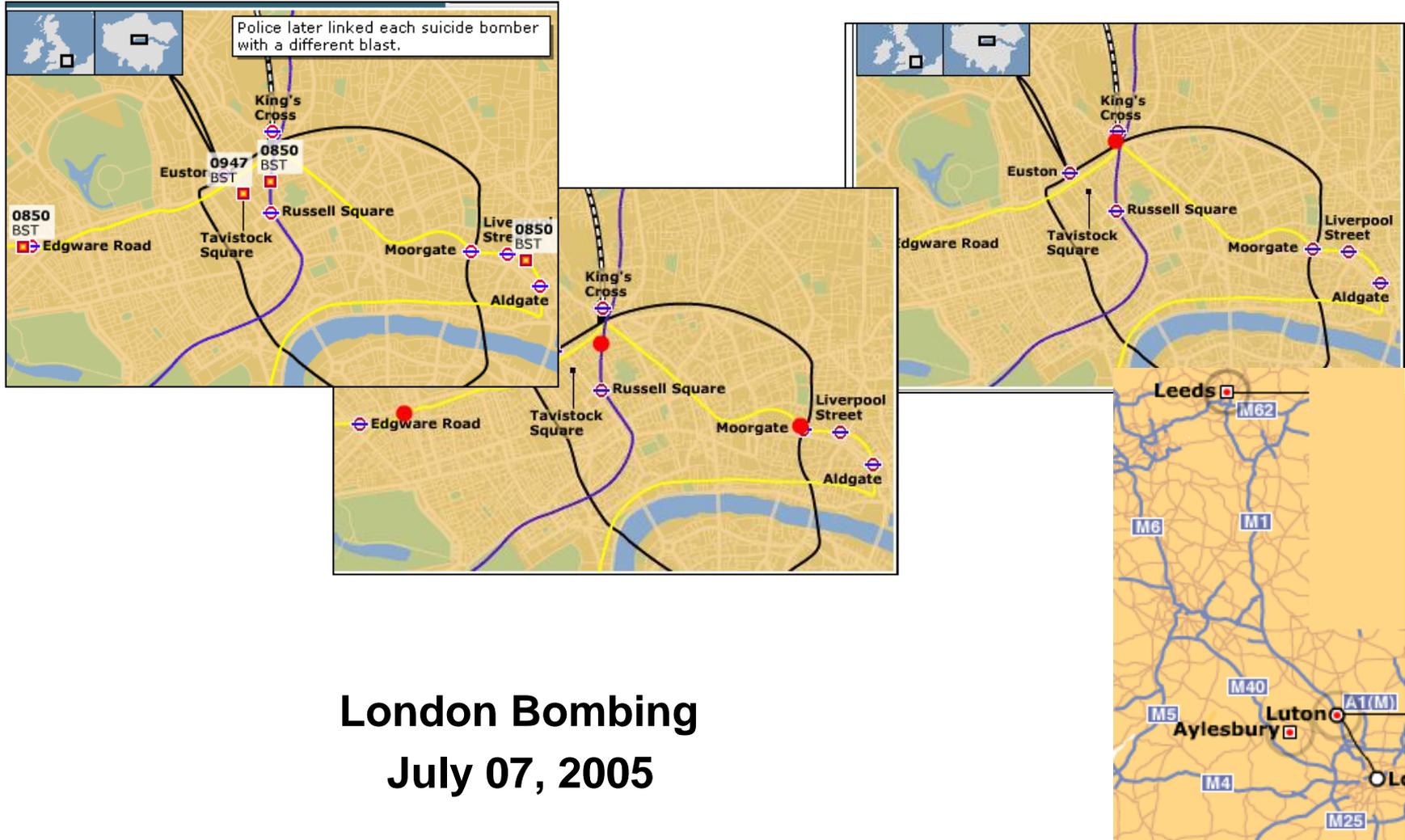
Query

Query Stamp	[Dropdown]	
Query Length	[Dropdown]	[Dropdown]
Query Relation	[Dropdown]	[Dropdown]

Close

- Convert the available temporal information into statements in Point-Interval Logic.
- Input these statements to **Temper** using the language editor of **Temper**.
- Construct a Point Graph representation of the set of Point Interval Logic (PIL) statements.
 - If the set of PIL statements is inconsistent then **Temper** will not be able to construct the Point Graph representation.
 - **Temper** will identify the subset of PIL statements causing the inconsistency.
 - User will remove the inconsistent statements.
- Once a consistent Point Graph has been constructed, it can be used to draw inferences.

- Knowledge Management and Reasoning
 - Forensics
 - Understanding of an incident of interest or a critical activity requires reconstruction of events that lead to an observable effect
 - Information regarding the incident/activity unfolds in no specific order and originates from different locations
 - Temporal information may be both qualitative and quantitative
 - Information may be inconsistent/incorrect
 - Information may contain hidden patterns or temporal relations that can help identify missing links
 - This calls for an automated tool for temporal knowledge representation, management, verification and reasoning
- **Temper** is also the temporal algorithm embedded in *Pythia*

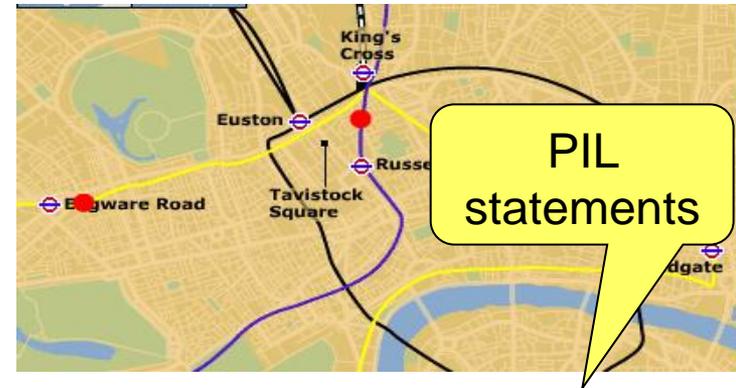


London Bombing July 07, 2005

Example: London Bombing



- There were four explosions in London.
- The sites of these explosions were: Travistock Square, Edgware Road, Aldgate and Russell Square.
- Three of these explosions (Edgware, Aldgate and Russell Square) were in trains.
- These trains left from King's Cross station. The journey of these trains ended in explosions.
- The time it takes a train from King's Cross to reach Edgware is at least 5 minutes.
- The time it takes a train from King's Cross to reach Aldgate is at least 4 minutes.
- The time it takes a train from King's Cross to reach Russell Square is at least 5 minutes.

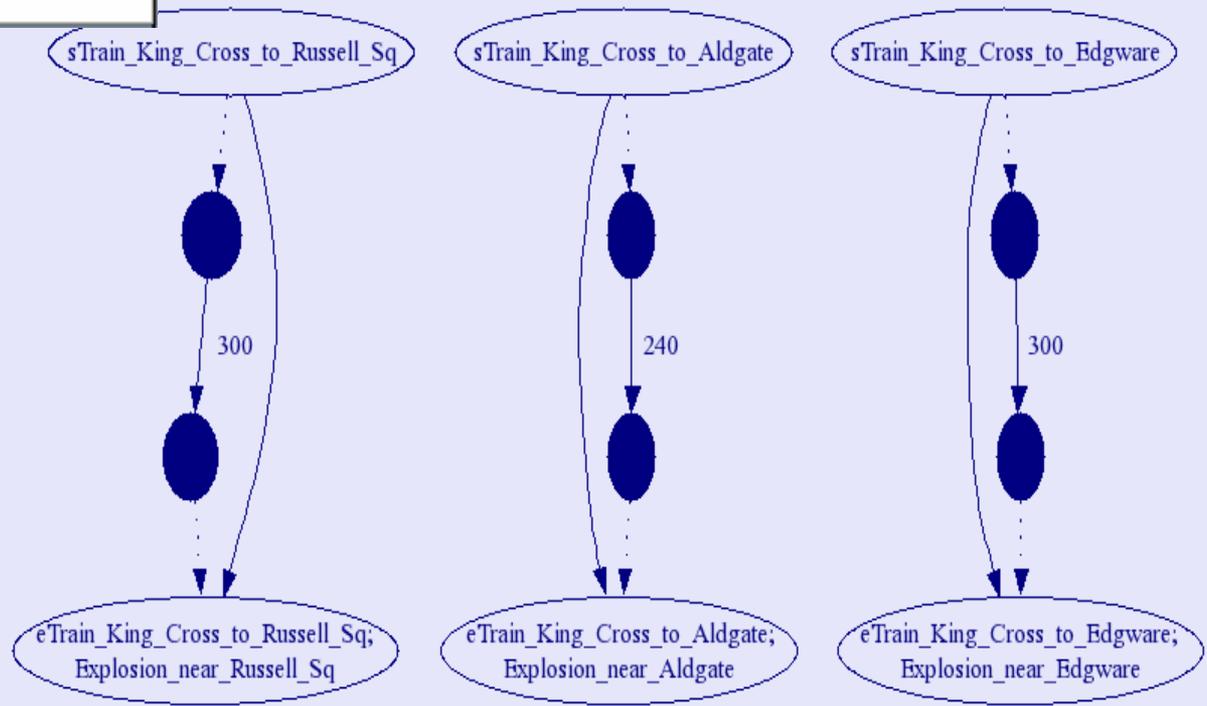


Interval Train_King_Cross_to_Edgware,
 Train_King_Cross_to_Aldgate,
 Train_King_Cross_to_Russell_Sq
Point Explosion_at_Travistock_Square,
 Explosion_near_Edgware,
 Explosion_near_Aldgate,
 Explosion_near_Russell_Sq
 Explosion_near_Edgware *finishes*
 Train_King_Cross_to_Edgware
 Explosion_near_Aldgate *finishes*
 Train_King_Cross_to_Aldgate
 Explosion_near_Russell_Sq *finishes*
 Train_King_Cross_to_Russell_Sq
Length [Train_King_Cross_to_Edgware] >= 0:5:0
Length [Train_King_Cross_to_Aldgate] >= 0:4:0
Length [Train_King_Cross_to_Russell_Sq] >= 0:5:0

- New
- Open ...
- Append ...
- Close
- Save ...
- Save As ...
- Recent Files
- Exit

Queries

Point Graph



PIL Statements

Compiled	To Be Deleted	Inferred	To Be Added	Comments
----------	---------------	----------	-------------	----------

```

sTrain_King_Cross_to_Edware < eTrain_King_Cross_to_Edware
sTrain_King_Cross_to_Aldgate < eTrain_King_Cross_to_Aldgate
sTrain_King_Cross_to_Russell_Sq < eTrain_King_Cross_to_Russell_Sq
Length [sTrain_King_Cross_to_Edware,eTrain_King_Cross_to_Edware] 300
Length [sTrain_King_Cross_to_Aldgate,eTrain_King_Cross_to_Aldgate] 240
Length [sTrain_King_Cross_to_Russell_Sq,eTrain_King_Cross_to_Russell_Sq] 300
Explosion_near_Edware f [sTrain_King_Cross_to_Edware,eTrain_King_Cross_to_Edware]
Explosion_near_Aldgate f [sTrain_King_Cross_to_Aldgate,eTrain_King_Cross_to_Aldgate]
Explosion_near_Russell_Sq f [sTrain_King_Cross_to_Russell_Sq,eTrain_King_Cross_to_Russell_Sq]

```

PIL statements

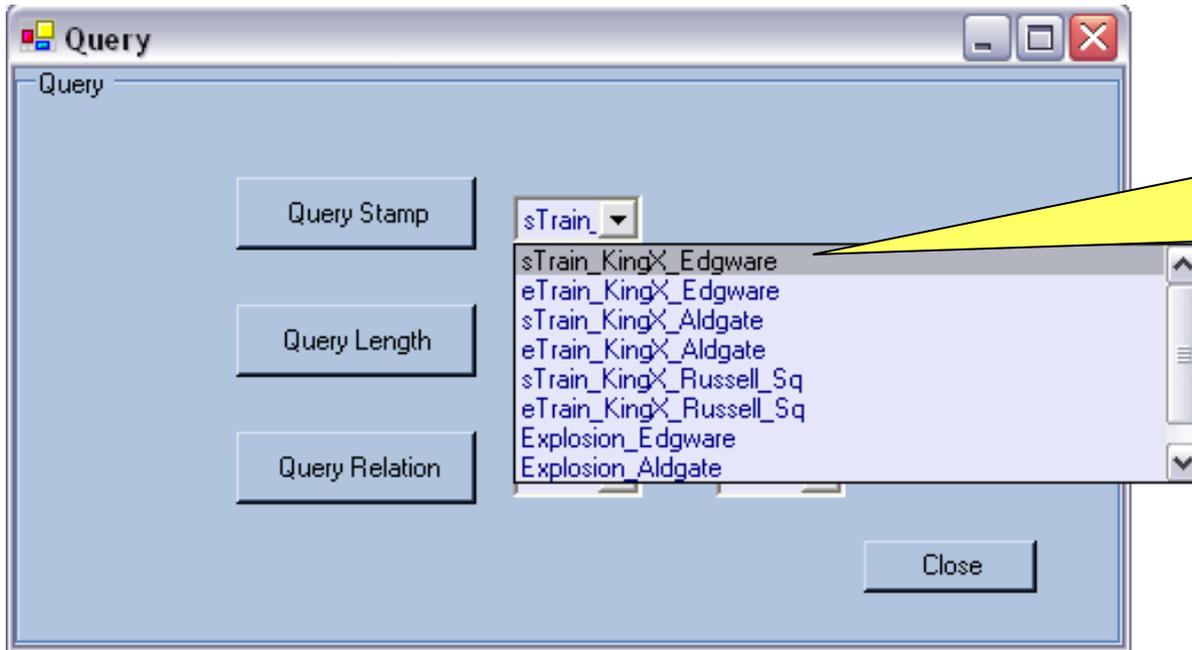
Activity Table **Real Time** Gantt Chart Output

```

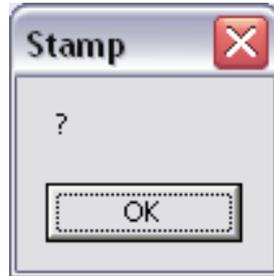
sTrain_King_Cross_to_Edware : 0 days 0 hour 0 minute 0 second
eTrain_King_Cross_to_Edware : 0 days 0 hour 0 minute 0 second
sTrain_King_Cross_to_Aldgate : 0 days 0 hour 0 minute 0 second
eTrain_King_Cross_to_Aldgate : 0 days 0 hour 0 minute 0 second
sTrain_King_Cross_to_Russell_Sq : 0 days 0 hour 0 minute 0 second
eTrain_King_Cross_to_Russell_Sq : 0 days 0 hour 0 minute 0 second
Explosion_near_Edware : 0 days 0 hour 0 minute 0 second
Explosion_near_Aldgate : 0 days 0 hour 0 minute 0 second
Explosion_near_Russell_Sq : 0 days 0 hour 0 minute 0 second
Explosion_at_Travistock_Square : 0 days 0 hour 0 minute 0 second

```

Example: London Bombing (cont'd)



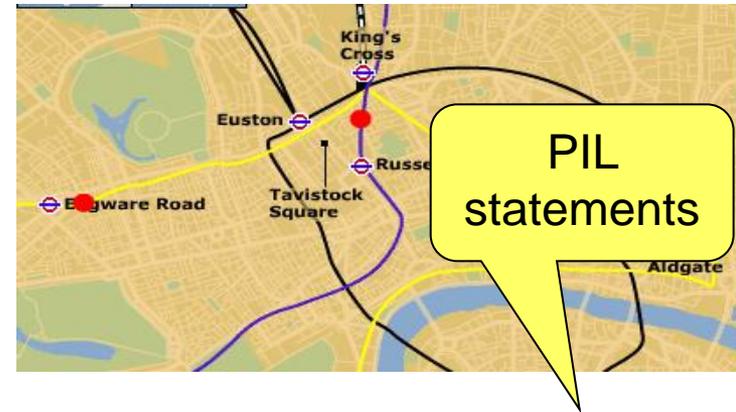
Query Stamp
(when did the train
to Edgware leave
from King's Cross?)



Example: London Bombing (cont'd)

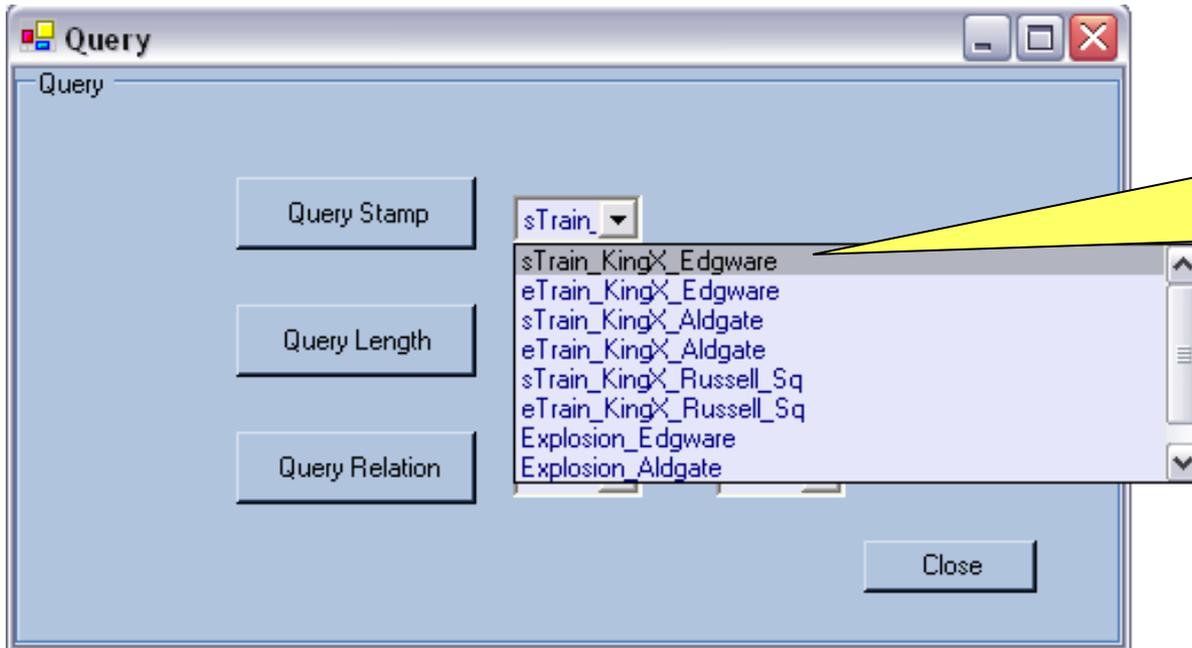


- The explosion near Edgware Road took place between time units 8:40 and 8:52.
- The explosion near Aldgate took place between time units 8:45 and 8:50.
- The explosion near Russell Square took place between time units 8:40 and 8:50.
- The explosion at Tavistock Square took place between time units 9:45 and 9:55.



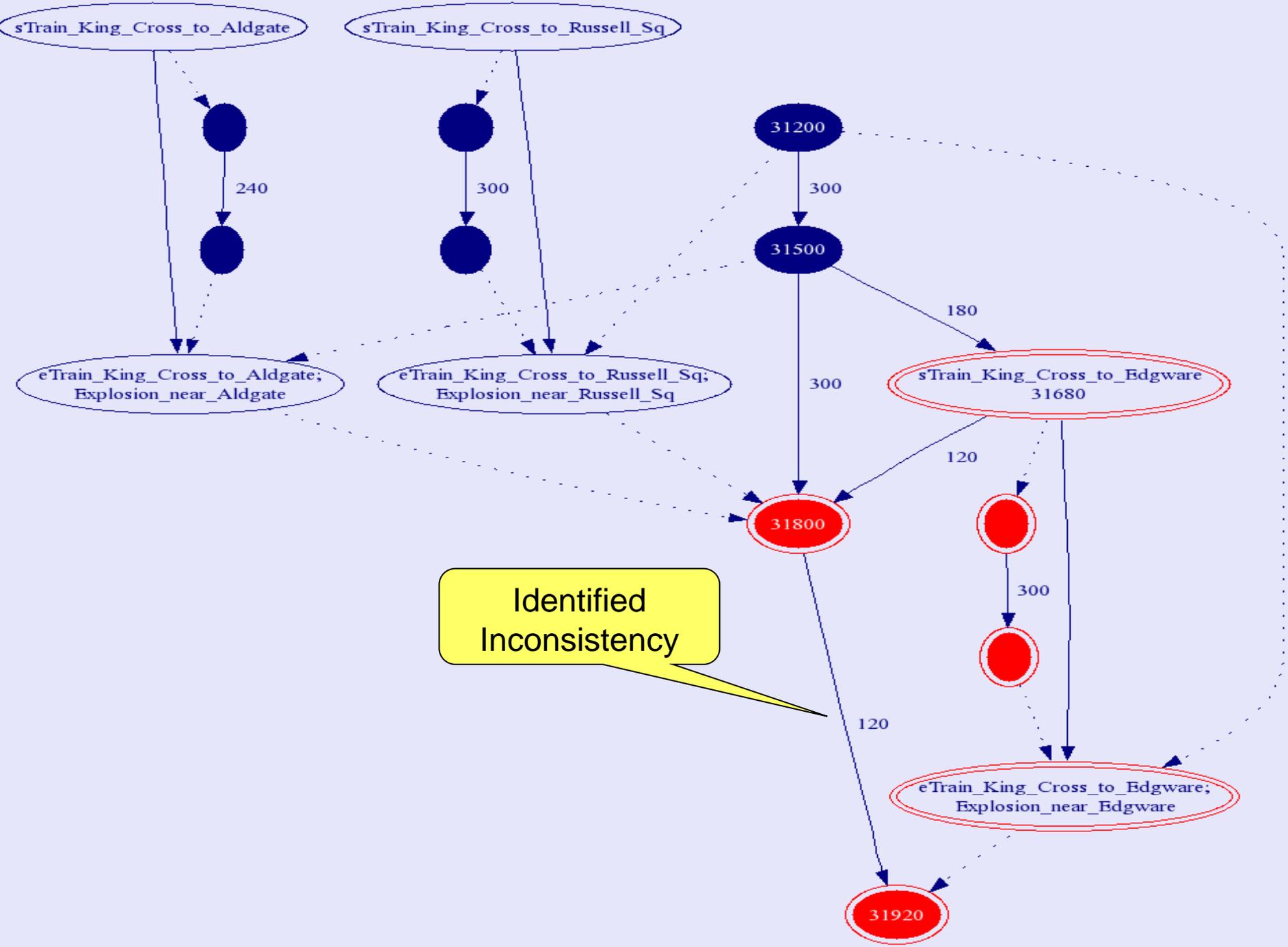
8:40 \Leftarrow *Stamp* [Explosion_near_Edgware] \Leftarrow 8:52
 8:45 \Leftarrow *Stamp* [Explosion_near_Aldgate] \Leftarrow 8:50
 8:40 \Leftarrow *Stamp* [Explosion_near_Russell_Sq] \Leftarrow 8:50
 9:45 \Leftarrow *Stamp* [Explosion_at_Tavistock_Square] \Leftarrow 9:55

Example: London Bombing (cont'd)



Query Stamp
(when did the train
to Edgbare leave
from King's Cross?)

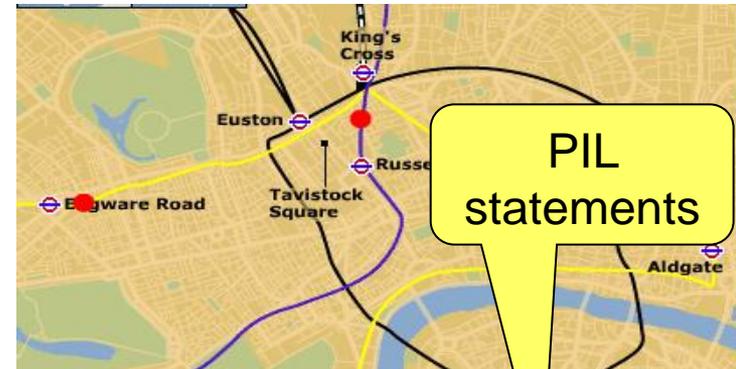




Example: London Bombing (cont'd)

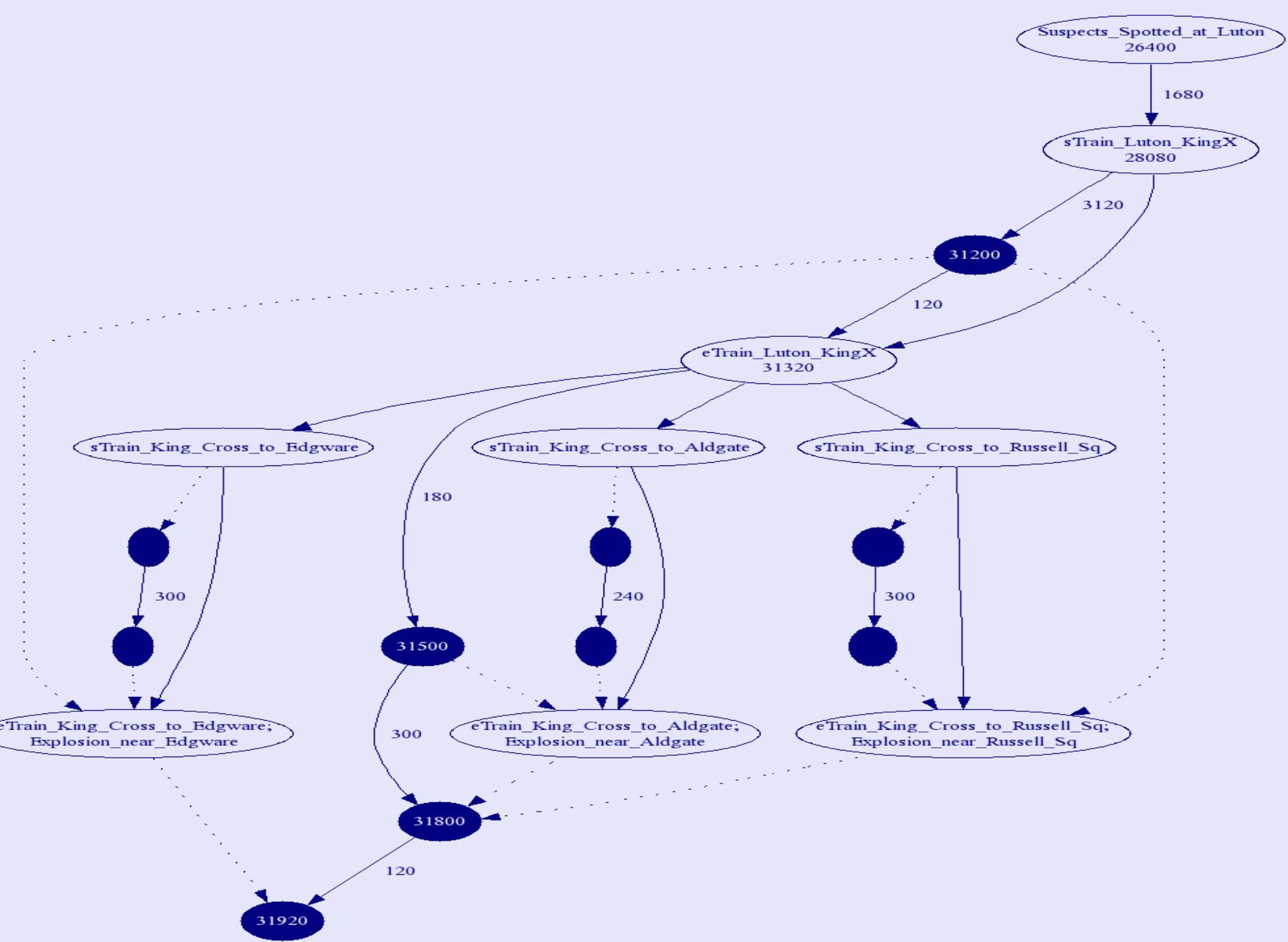


- The alleged four bombers spotted entering the Luton station at time unit 7:20.
- The next train from Luton to King's Cross left at 7:48 reaching King's Cross at 8:42.
- Train to Edgware left after the train from Luton.
- Train to Aldgate left after the train from Luton.
- Train to Russell Sq. left after the train from Luton.

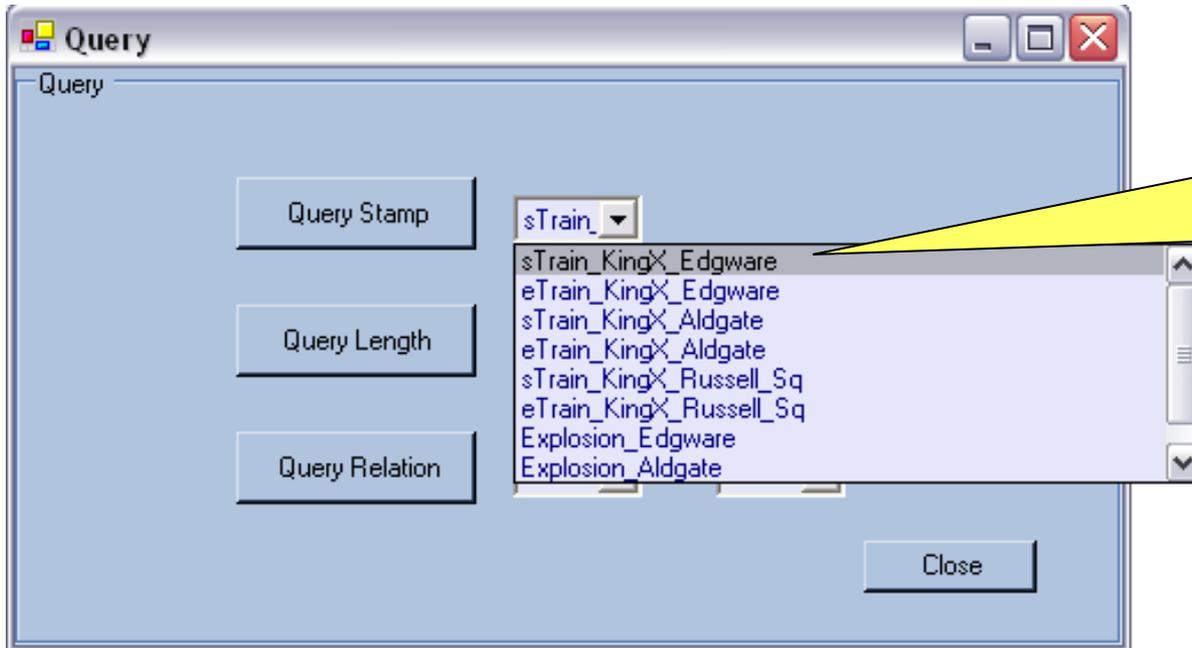


Interval Train_Luton_to_King_Cross
Point Bombers_spotted_at_Luton
Stamp [Bombers_spotted_at_Luton] = 7:20
Stamp [sTrain_Luton_to_King_Cross] = 7:48
Stamp [eTrain_Luton_to_King_Cross] = 8:42
eTrain_Luton_to_King_Cross before
 Train_King_Cross_to_Edgware
eTrain_Luton_to_King_Cross before
 Train_King_Cross_to_Aldgate
eTrain_Luton_to_King_Cross before
 Train_King_Cross_to_Russell_Sq





Example: London Bombing (cont'd)



Query Stamp
(when did the train
to Edgware leave
from King's Cross?)





- **A formal approach to modeling and analyzing temporal information related to an event of interest, e.g., terrorist acts**
- **A software implementation of the approach with**
 - **An easy-to-use input language**
 - **Analysis toolkit that includes a consistency checker and a reasoning tool with a query language/interface**
 - **An efficient revision mechanism that helps add/modify temporal information without restarting the whole process**
 - **A graphical interface**
- **What might be added in future**
 - **Connectivity to temporal information in databases**
 - **Automated extraction of temporal information from textual source(s)**
 - **Better user/analyst input/output interfaces for display of information (both input and inferred)**



- Integration of the three dimensions of spatial knowledge with the temporal dimension to create a unified approach for handling change
- Exploring equivalence/overlaps between temporal logic operators and temporal relations of **Temper** for enhancements in input/query languages
- Development or addition of sophisticated GUI for inputs and outputs, e.g., something similar to WebTAS APIs