



IBM T.J. Watson Research Center

Automatic Composition of Dataflows using Semantic Graph-Transformation Models of Components

**Zhen Liu,
Anand Ranganathan,
Anton Riabov**

Dataflows in Information Processing Systems

- Specifies the flow of data from one or more sources and through a variety of components
- Finally produces results that are requested by some user or application.
- Different kinds of data
 - Unstructured, semi-structured or structured,
 - Different formats including text, video, audio and images.
- Examples: multimedia processing and delivery networks, stream-processing systems, sensor networks
- In this work, dataflow is in the form a DAG (Directed Acyclic Graph)

Automatic Composition of Dataflows

■ Motivation

- Very difficult for users to compose the dataflow graph manually
- Very large number of data sources and components
- Set of available sources and components can change dynamically
- Automatic Composition simplifies user interaction

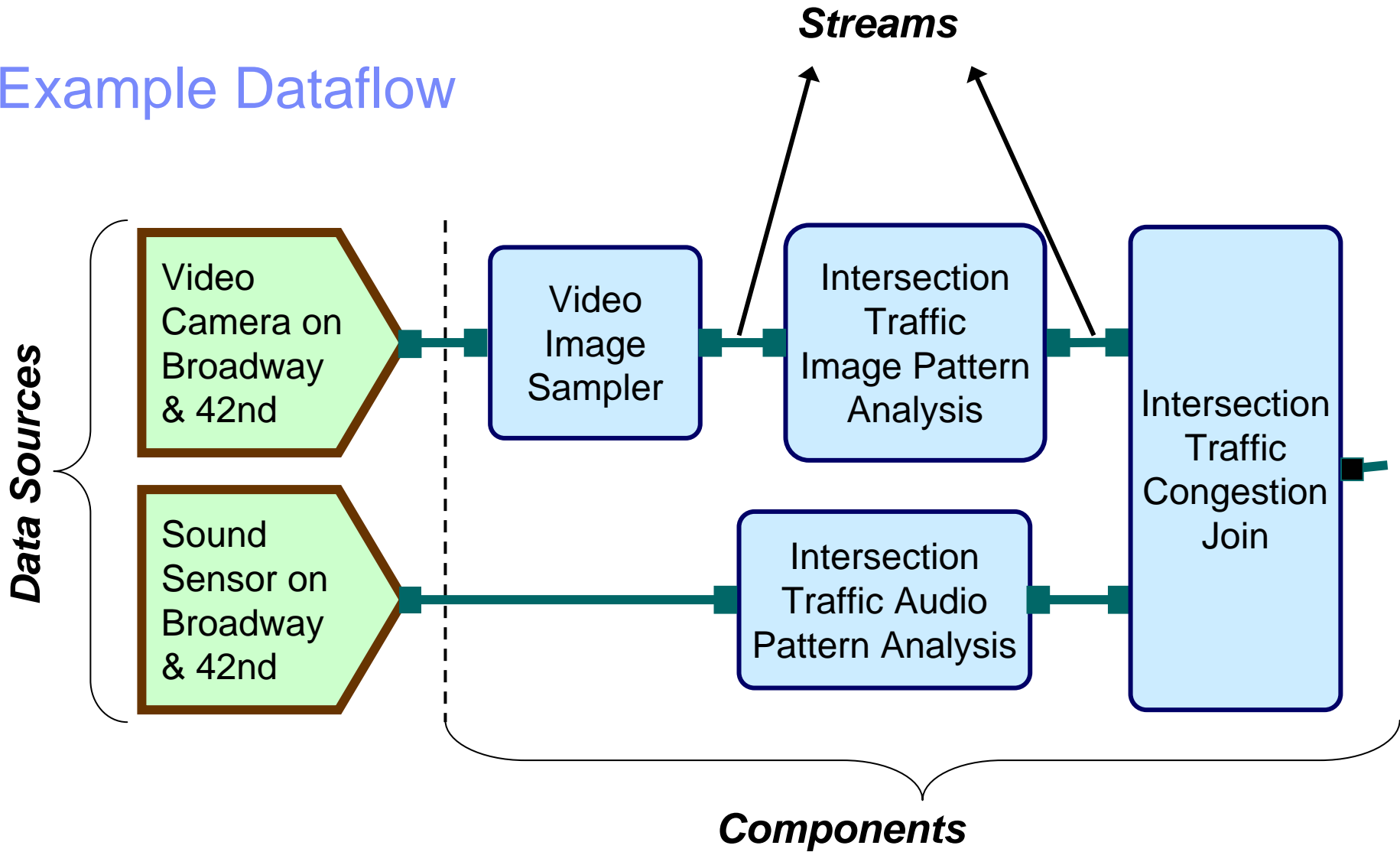
■ Input to Problem:

- Properties of desired data as expressed in a user query
- Set of components with descriptions of inputs and outputs

■ Output of Problem:

- Dataflow graph describing flow of data through different components in a system to produce desired output

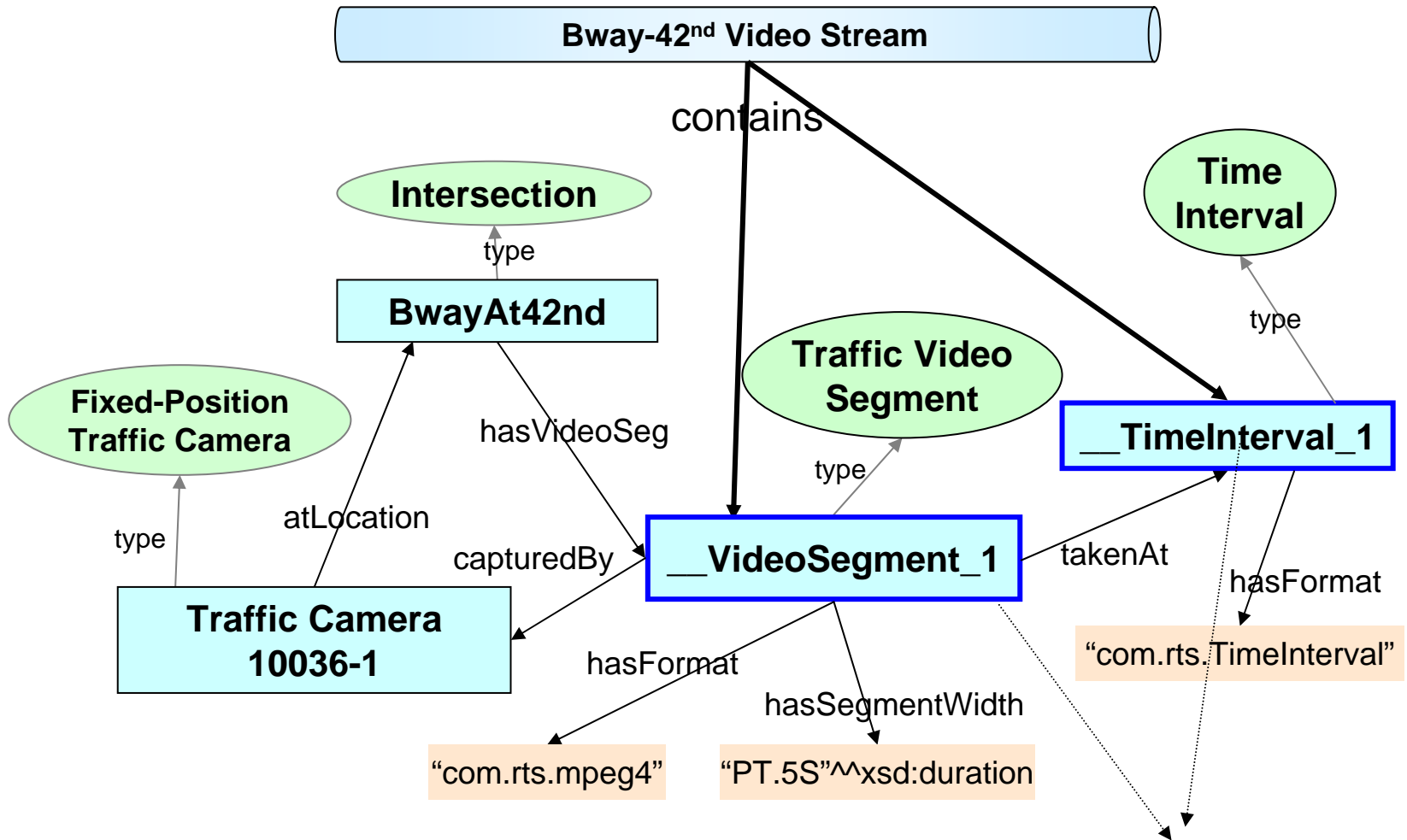
Example Dataflow



Overview of Approach

- **Expressive descriptions of streams and of inputs and outputs of components**
 - More Expressive than OWL-S
 - Based on Instance-level graph patterns
 - Allows use of variables for propagating semantics from input to output
- **Derive conditions for connecting a stream to a component**
- **User queries expressed in SPARQL**
 - PRODUCE ?cl, ?time
WHERE (?cl rdf:type CongestionLevel) , (?time rdf:type Time) ,
(?cl ofLocation BwayAt42nd) , (?cl atTime ?time)
- **Use of AI Planning techniques for constructing the dataflow given query and description of components**
 - Offline DLP reasoning used to infer additional facts about outputs

Example Description of Stream

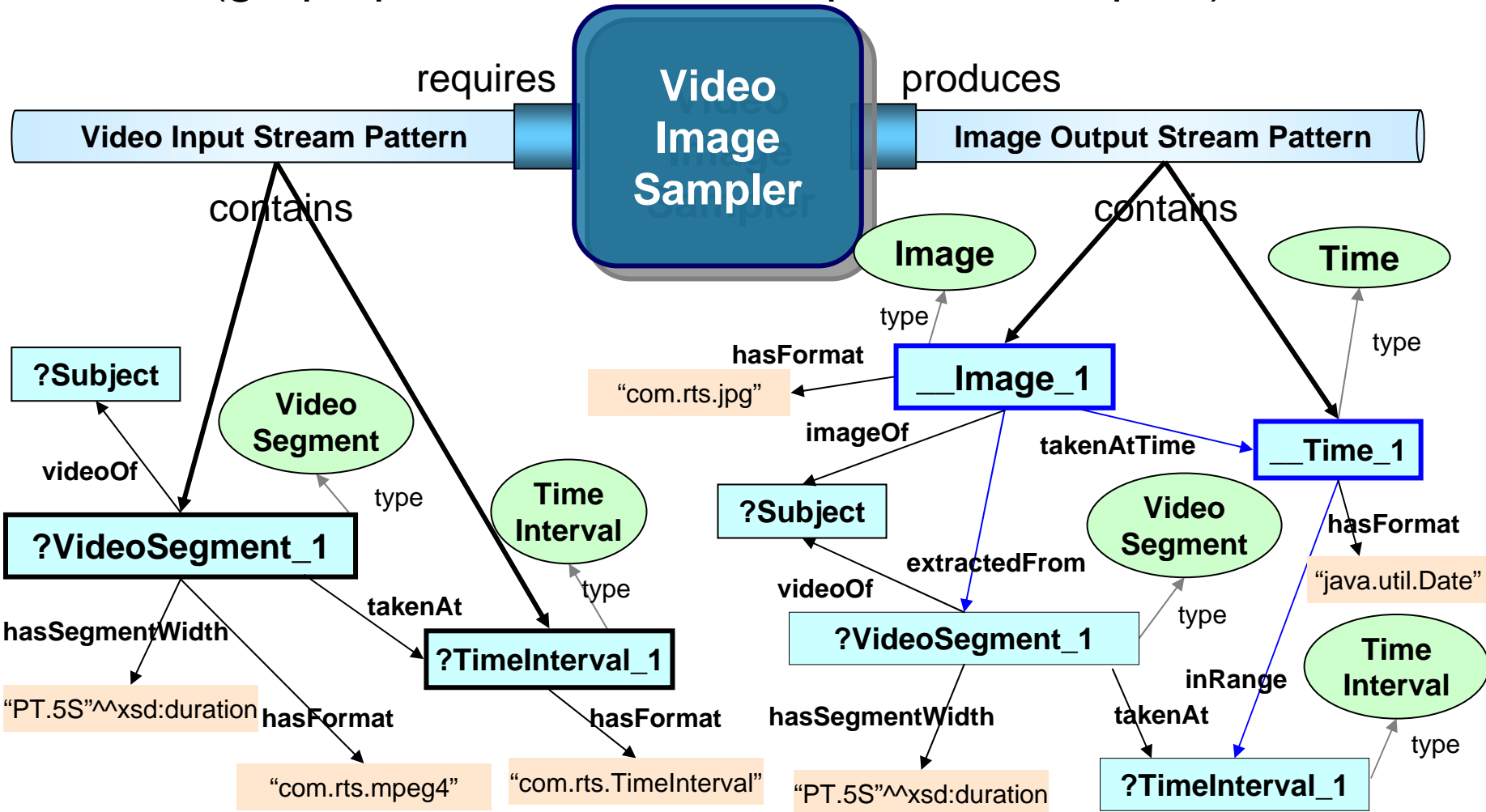


Exemplar represents the data elements in a typical packet that may vary across different packets

Stream Model

- Description of stream is in terms of the description of an example packet on the stream
- Each stream is of the form (E, R)
 - E is the set of data elements in an example packet on the stream
 - R is an RDF graph describing properties of the data elements
- Different packets in stream have different values of the data elements
 - Elements in E are represented as **exemplar individuals** and **exemplar literals**
 - Abstract over the specific data values in different packets
 - Exemplar individual is an OWL individual that belongs to a special class *Exemplar*
 - Path in URI starts with “__”
 - Exemplar literal represented as a Unicode string with initial “__”
- Exemplars are treated as existentially quantified variables
 - Different packets in stream substitute different values for them

Example Description of Component (graph patterns describe inputs and outputs)



Component Model

- Each input message requirement and output message produced is described as a message pattern of the form
$$MP = (VS, GP)$$
 - *VS* is a set of variables or exemplars representing data objects contained in the message
 - *GP* is an RDF graph pattern describing required semantics of data elements
 - Variables are elements in input message, which may be carried to output message
 - Exemplars are new objects created in output message
- Each component is a set of input and output patterns
- Component can be described as a **graph transformation**
 - New output stream generated from input stream by glueing in new vertices and edges in output graph pattern and removing any vertices and edges that don't appear in output pattern, but that are present in input pattern.

Conditions for connecting a stream as input to a component

- Input message pattern is $MP = (VS, GP)$
- Stream is $S = (E, R)$
- S matches MP iff there exists a variable substitution θ , such that
 - $\theta(VS) \subseteq E$, i.e. stream contains all required elements
 - $R \cup O \models \theta(GP)$, where O is a domain ontology and \models is an entailment relation between RDF graphs, i.e. semantic description of stream satisfies all constraints expressed in input graph pattern