Context-Aware Computing 2

Context Representation and Reasoning

Mobile Computing

Minho Shin

2012. 11

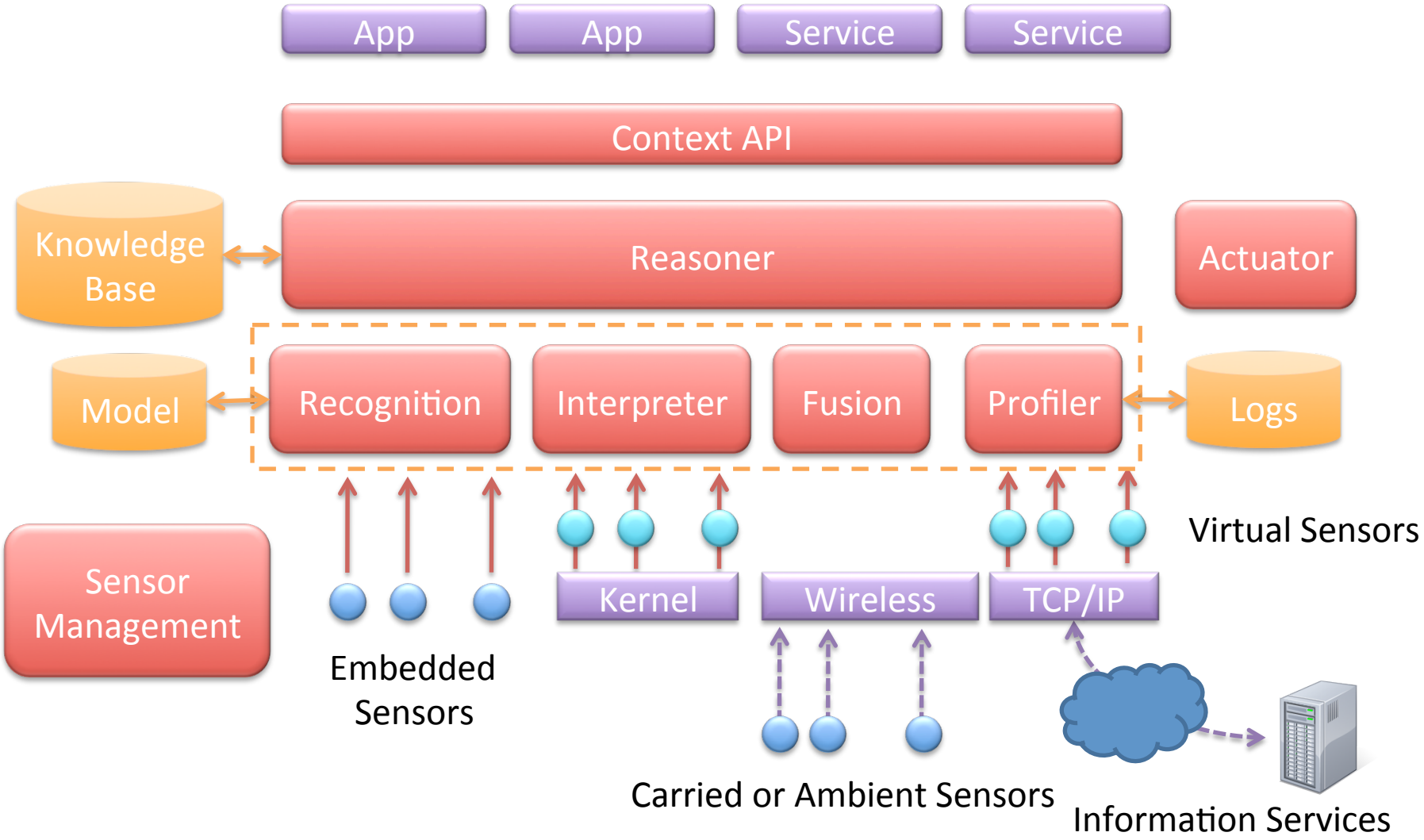
Definitions

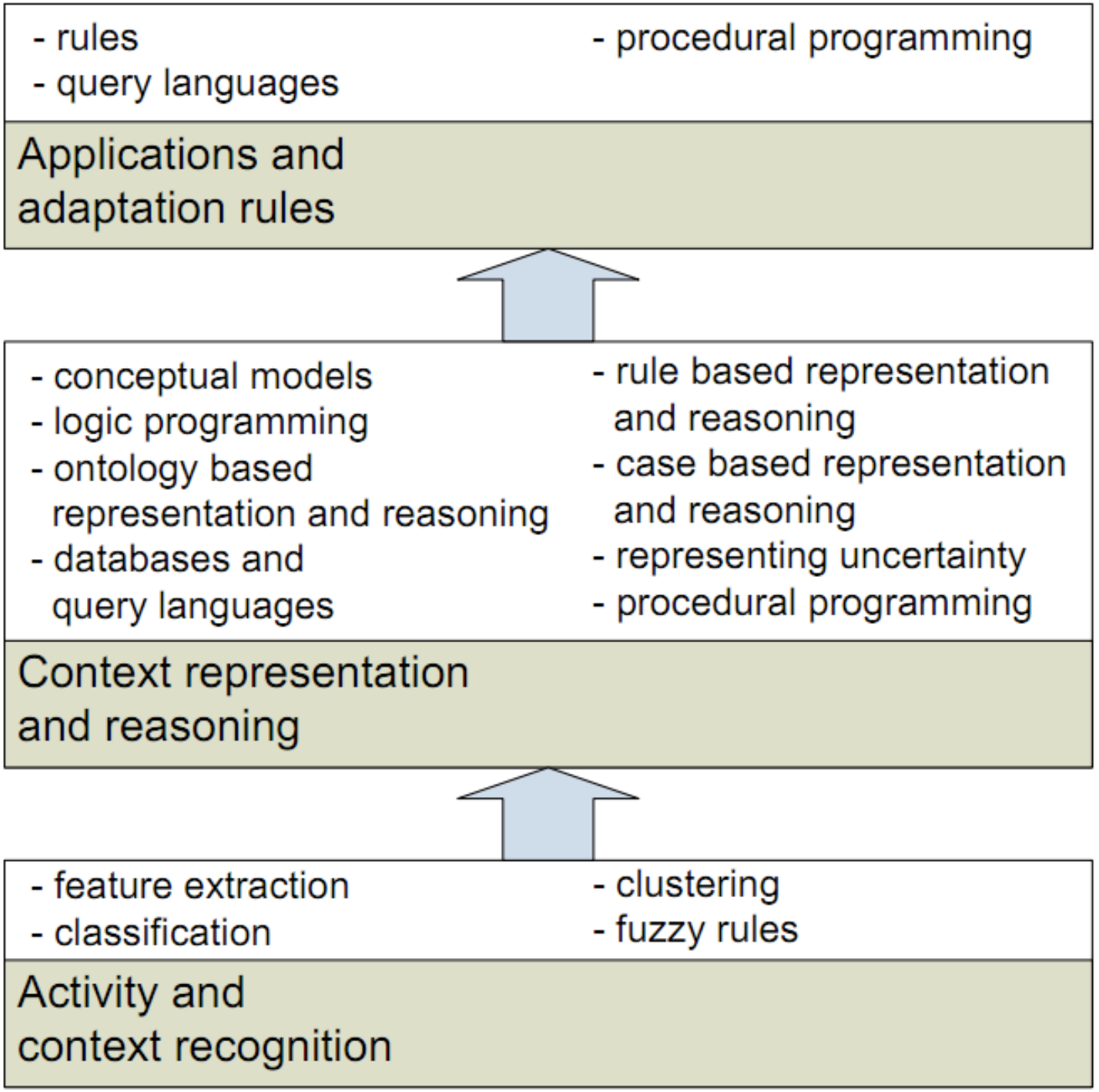
- Dey
  - Context is any information that can be used to characterize the situation of an entity.
  - An entity is a person, place, or object that is considered relevant to the interaction
  - between a user and an application, including the user and applications themselves

Context Awareness

- Computers can both sense, and react based on their environment.
- Context aware systems are concerned with
  - acquisition of context (e.g. using sensors to perceive a situation),
  - abstraction and understanding of context (e.g. matching a perceived sensory stimulus to a context),
  - application behavior based on the recognized context (e.g. triggering actions based on context).

System Architecture for Context-Aware Computing




- rules
- query languages
- procedural programming
Applications and adaptation rules
- conceptual models
- logic programming
- ontology based representation and reasoning
- databases and query languages
- rule based representation and reasoning
- case based representation and reasoning
- representing uncertainty
- procedural programming
Context representation and reasoning
- feature extraction
- classification
- clustering
- fuzzy rules
Activity and context recognition

Context Representation & Reasoning

- How can we make inferences on the user context based on lower context data?
  - Logic programming
  - Ontology-based
  - Case-based
- Knowledge Representation & Reasoning

LOGIC PROGRAMMING FOR CR&R

Logic Programming Approach

- Express variety of contexts by first order logic

Background: First-order logic

- Propositional Logic
  - No quantifiers ($\exists$ or $\forall$)
  - $p \rightarrow q$
- First-order logic
  - Statement expressed as a function: P(x)
    - Human(x)
  - Quantifiers allowed
    - $\forall$*people x P(x)* $\rightarrow$ *Q(x)*
  - No predicates or functions as arguments
- Higher-order logic
  - Predicates or funcations can be arguments

Background: First-order logic

- Propositional Logic
  - No quantifiers ($\exists$ or $\forall$)
  - $p \rightarrow q$
- First-order logic
  - Statement expressed as a function: P(x)
    - Human(x)
  - Quantifiers allowed
    - $\forall$*people x P(x)* $\rightarrow$ *Q(x)*
  - No predicates or functions as arguments
- Higher-order logic
  - Predicates or functions can be arguments

Context Model

- Represent context as first-order predicates
- Name of predicate is type of context
  - Location(…), Temperature(…), Time(…), …
- Examples:
  - *Location (chris, entering, room 3231)*
  - *Temperature (room 3231 , ‘‘=’’, 98 F)*
  - *Sister (venus, serena)*
- Arguments in Subject-Verb-Object (SVO) format
  - *ContextType(<subject>, <verb>, <object>)*
  - *Location (<person or object>, {entering, leaving, in}, <location>)*
- Each argument can take any types: string, integer, list, …
- Each context type corresponds to a class in ontology

Operations on Contexts

- Contruct complex context expressions by Boolean operations:
  - Conjunction($\wedge$), Disconjunction($\vee$), Negation($\neg$)
- Examples
  - *Location (Manuel , Entering, Room 3211)*
    $\wedge$ SocialActivity(Room 3211, Meeting)
    - "Manuel is entering Room 3211 and that there is a meeting going on in that room."
  - *Lighting(Room 3234, Off) $\vee$ Lighting(Room 3234, Dim)*
    - "The lighting in Room 3234 is either off or dim."
  - *$\neg$Location (Manuel , In, Room 3211)*
    - "Manuel is not in Room 3211."

Existential & Universal Quantification

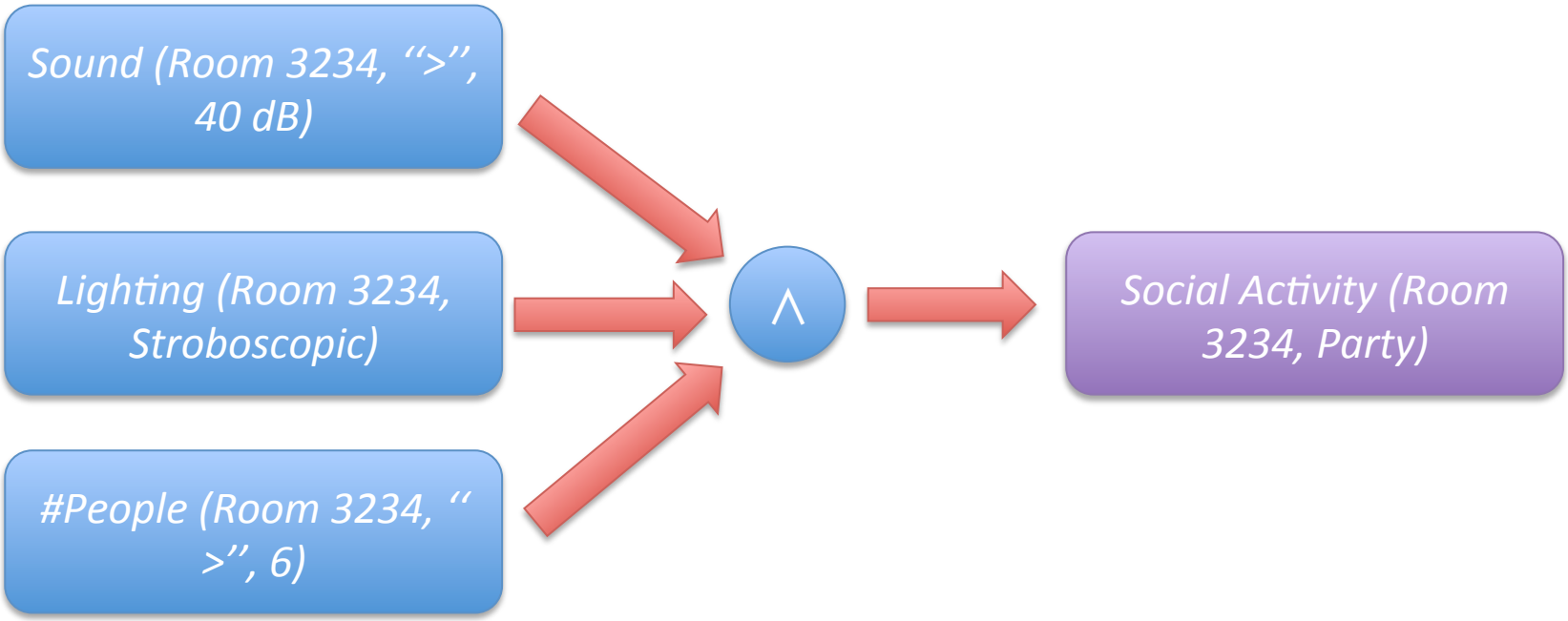
- Existential quantifier (“there exists”)
  - True for at least one value
  - $\exists_{Location}\, y$ *Location(Chris, In, y).*
    - “Chris is in some location”
- Universal quantifier (“for all”)
  - true for all values
  - $\forall_{People}\, x$ *Location(x, In, Room 3231)*
    - all people are in room 3231

Rules

- Can derive a new context from other (sensed) contexts
  - <existing context> => <new context>
  - *Sound (Room 3234, ‘‘>’’, 40 dB) ∧ Lighting (Room 3234, Stroboscopic) ∧ #People (Room 3234, ‘‘ >’’, 6) =>Social Activity (Room 3234, Party)*
  - “there is a party going on in a room if the level of sound in the room is high, stroboscopic lights are on and the number of people in the room is greater than some threshold value.”

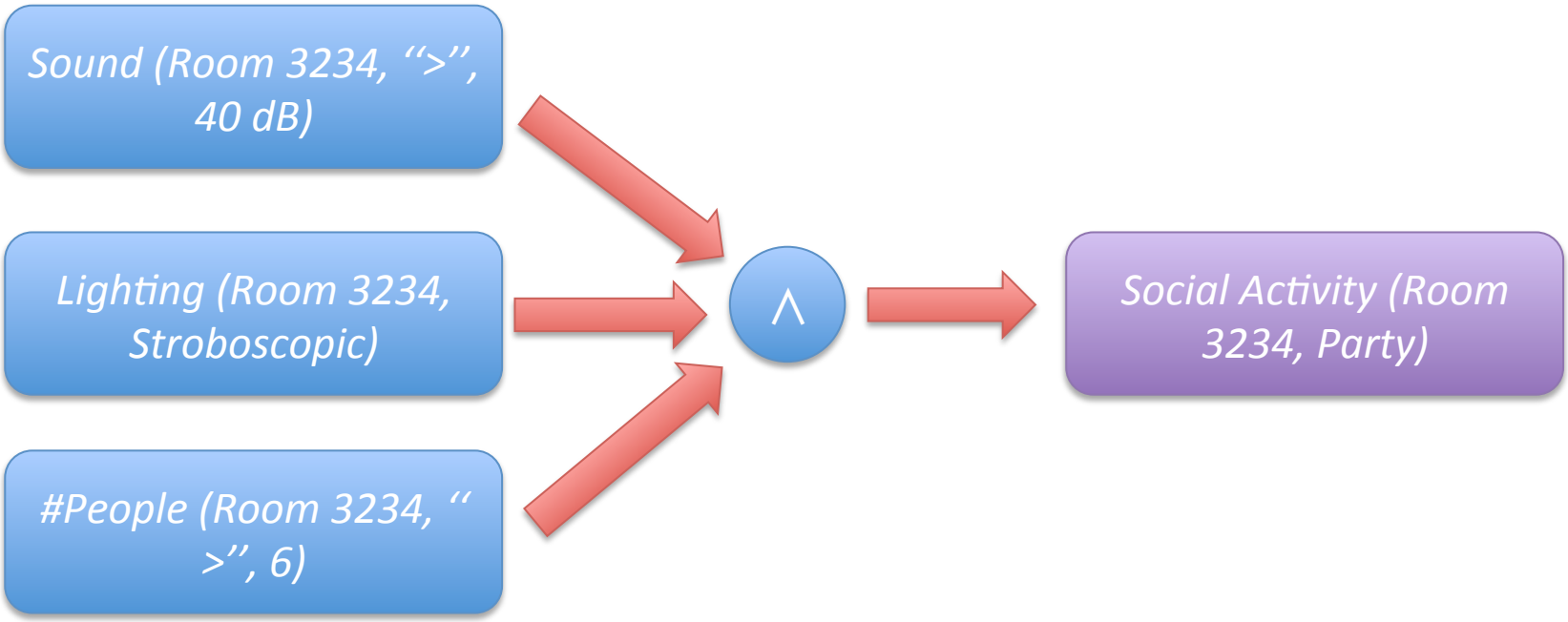
Rules

- Derive a new context from other contexts
  - <existing context> => <new context>



Rules

"there is a party going on in a room if the level of sound in the room is high, stroboscopic lights are on and the number of people in the room is greater than some threshold value."



Methods for Providing Context

- How can an application obtain a context of interest from the *context engine*?
- Context Query
  - Application sends a query to the context engine
- Subscribe-Notify
  - Subscribe for certain contexts and get notified whenever that context becomes true.

Methods for Providing Context

- Context Query
  - Application sends a query to the context engine
  - In a query, one or more arguments of the predicate is replaced by a variable (say X) to indicate that the application would like values of the variable to make the predicate true.
  - Ex: “Who is in room 3231?”
    - Query: *Location(X, in, room 3231)*

Methods for ProvidingContext

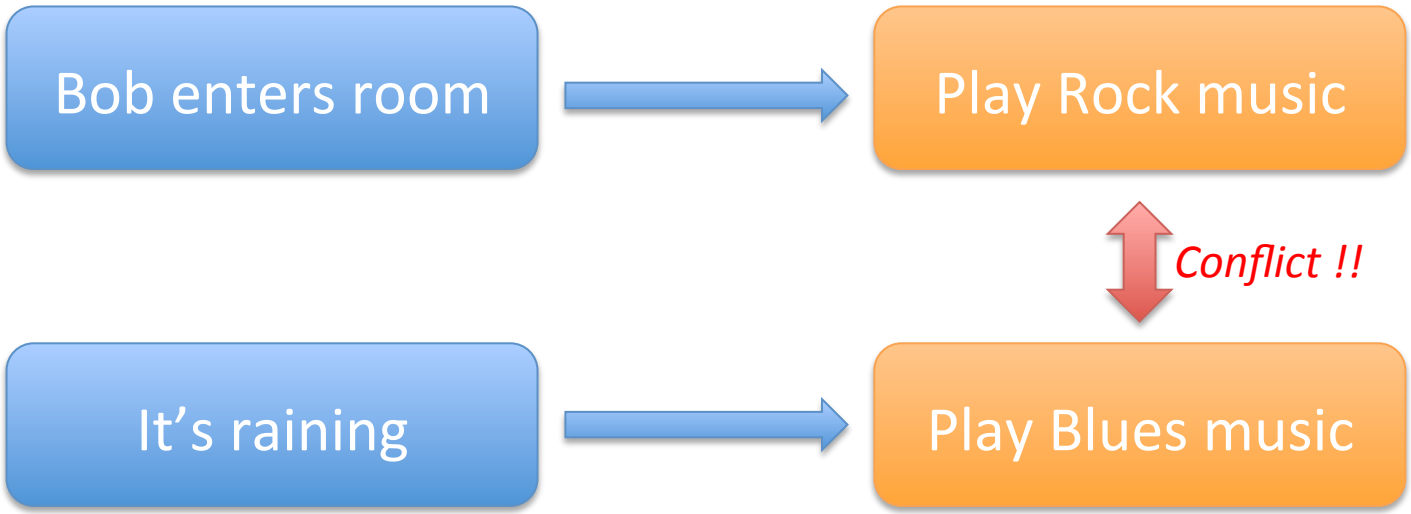
- Subscribe-Notify
  - Subscribe for certain contexts and get notified whenever that context becomes true.
  - For example, an application wants to receive notifications when outside temperature is below 5°C
    - Temperature(outside, “<“, 5°C)

Context-Sensitive Behavior

- A context-aware jukebox App could specify that whenever Bob enters his room, the playMusic() method must be invoked.
- Configuration file defines rules that associate certain contexts with an action

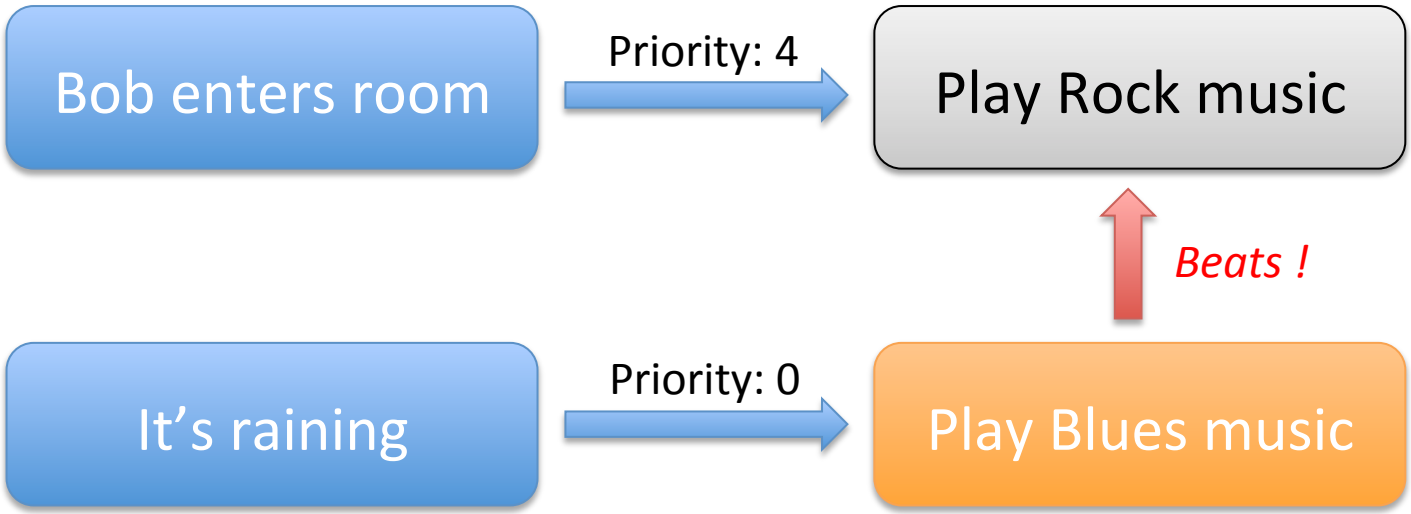
Challenge: Conflicts

- Two rules may cause a conflict in actions



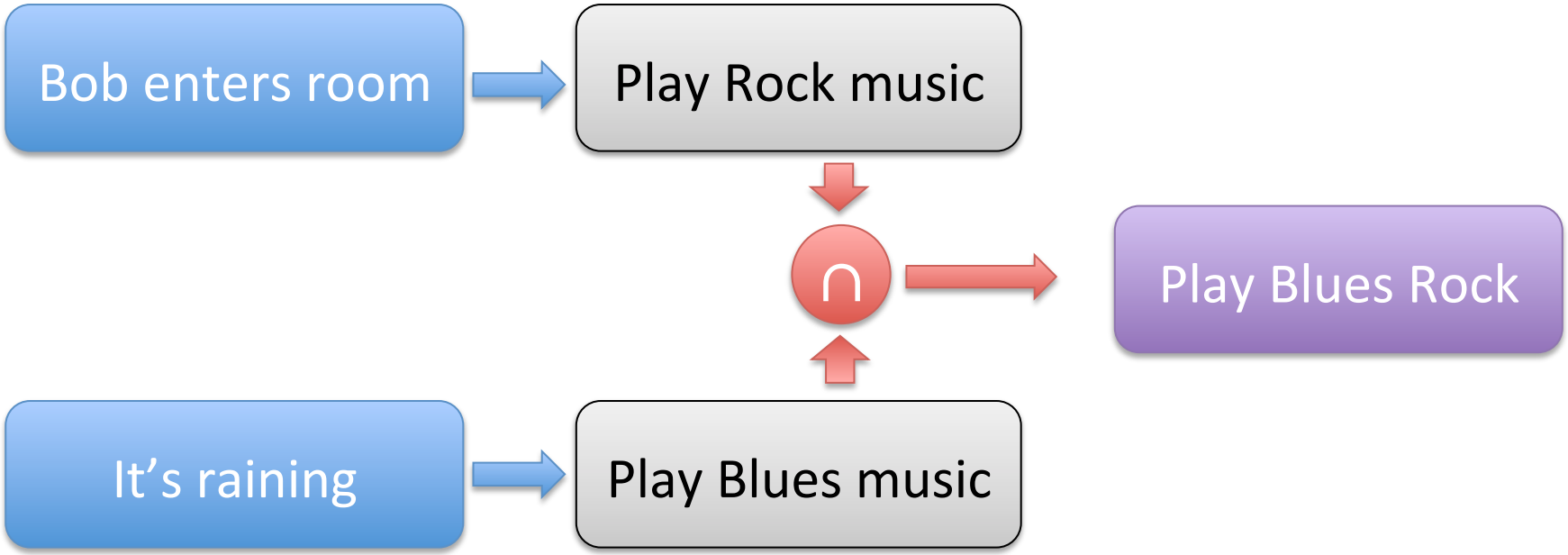
Conflict Resoultion: Prioritized Rule

- Set priority to each rule



Conflict Resoultion: Compromise

- Come up with a compromised action (find intersection of two actions)



Conflict Resoultion: Learning

- Choose an action and get feedback from the user
- Or let the user choose?
- Then learn user's preference between conflicting actions

Challenge: Context Evaluation

- Suppose we have *n* rules in total
  - Either context-synthesize rules or action rules
- For each small changes in sensor values or lower contexts, we need to evaluate *n* rules, i.e., *n* contexts: O(n)
- How can we minimize the number of necessary context evaluation?

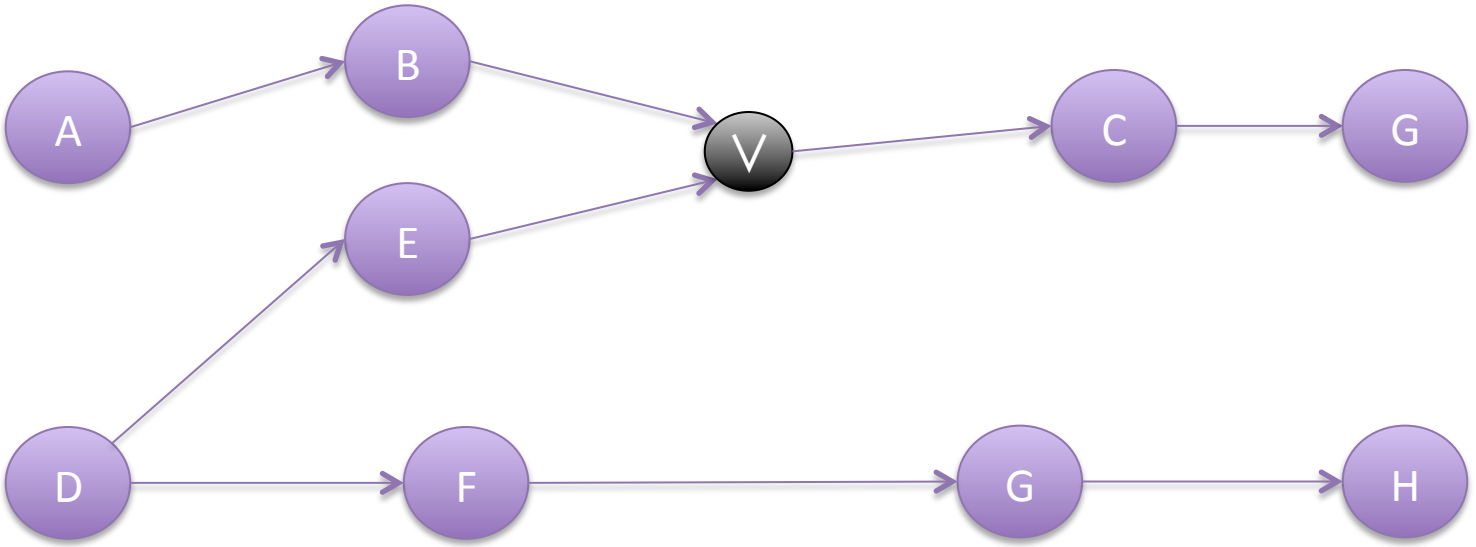
Optimization

- Optimize context evaluation

- If Context-A is true, then Context-B is true
- contrapositive:
- If Context-B is not, then Context-A is not
- So, if Context-B is found false, we don't need to evaluate Context-A (false)

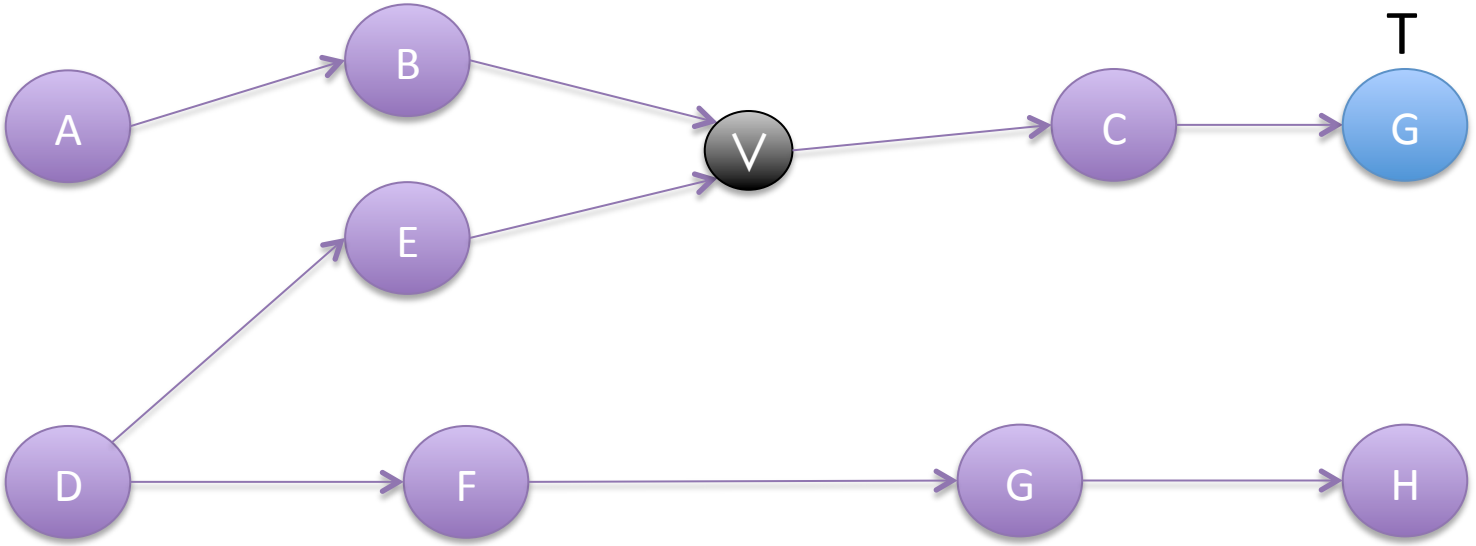
POSET (Partially Ordered Set)

- With contexts, form a poset and evaluate from the right to left until false



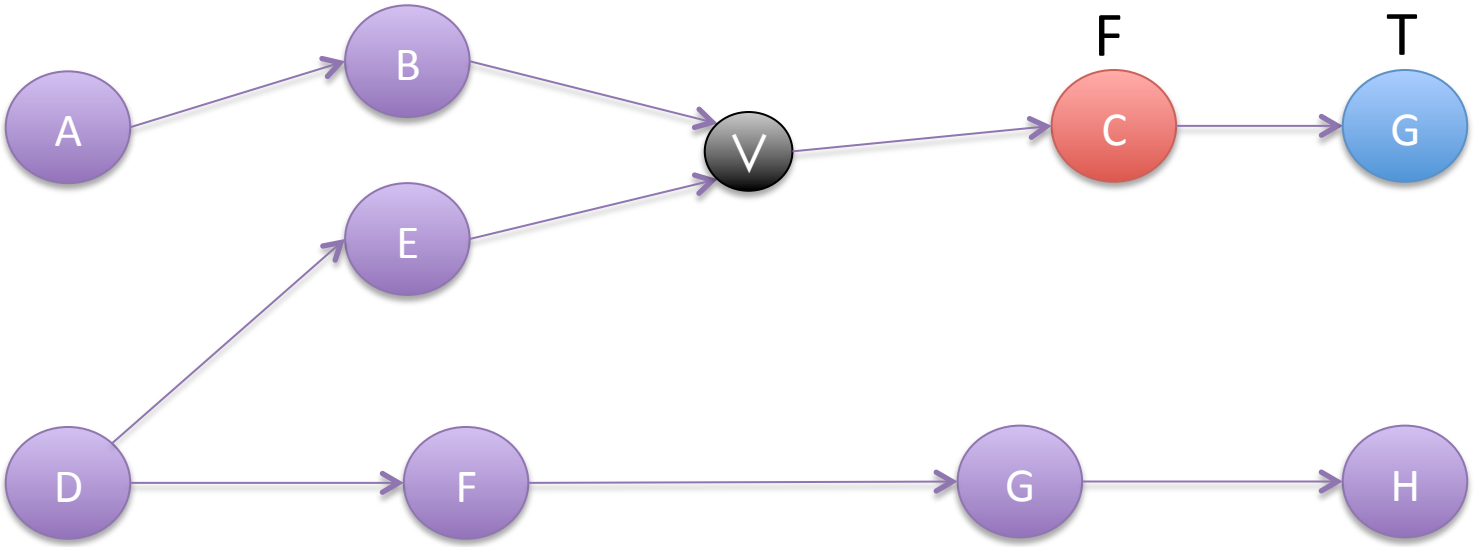
POSET (Partially Ordered Set)

- With contexts, form a poset and evaluate from the right to left until false



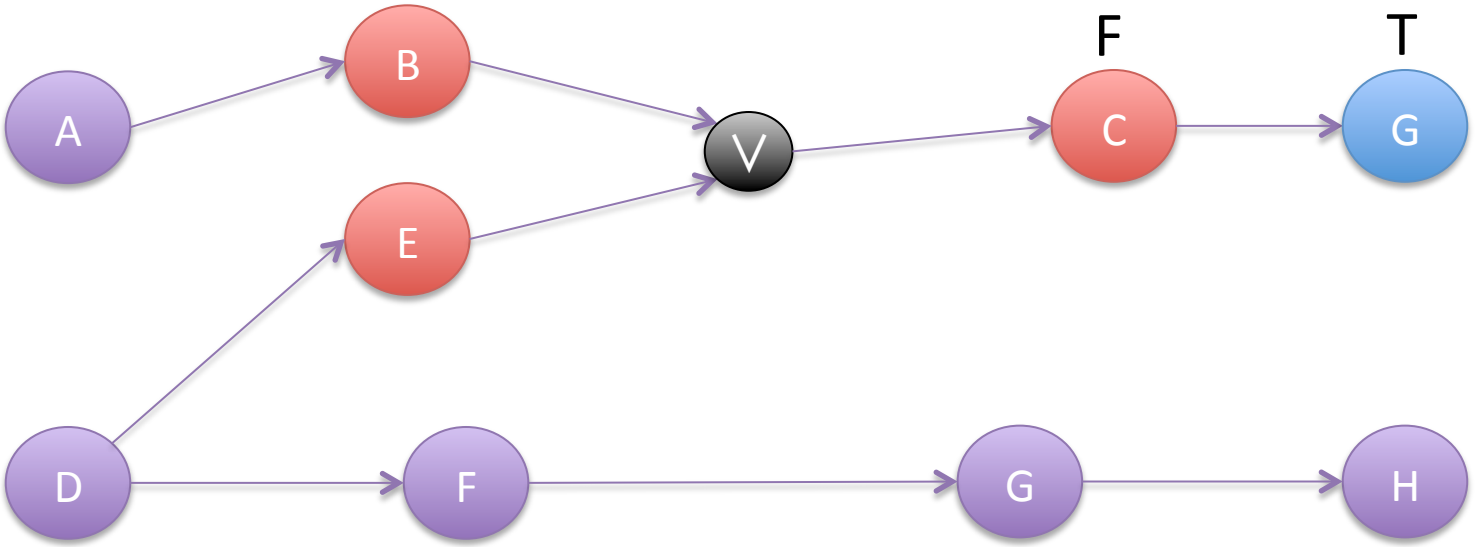
POSET (Partially Ordered Set)

- With contexts, form a poset and evaluate from the right to left until false



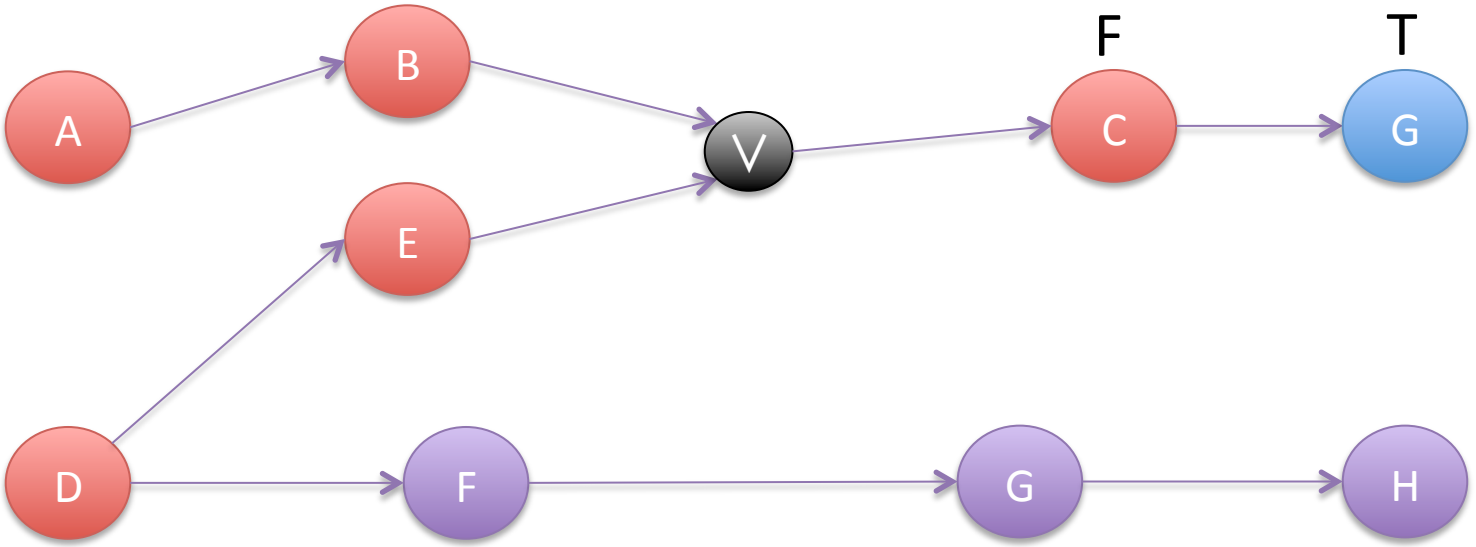
POSET (Partially Ordered Set)

- With contexts, form a poset and evaluate from the right to left until false



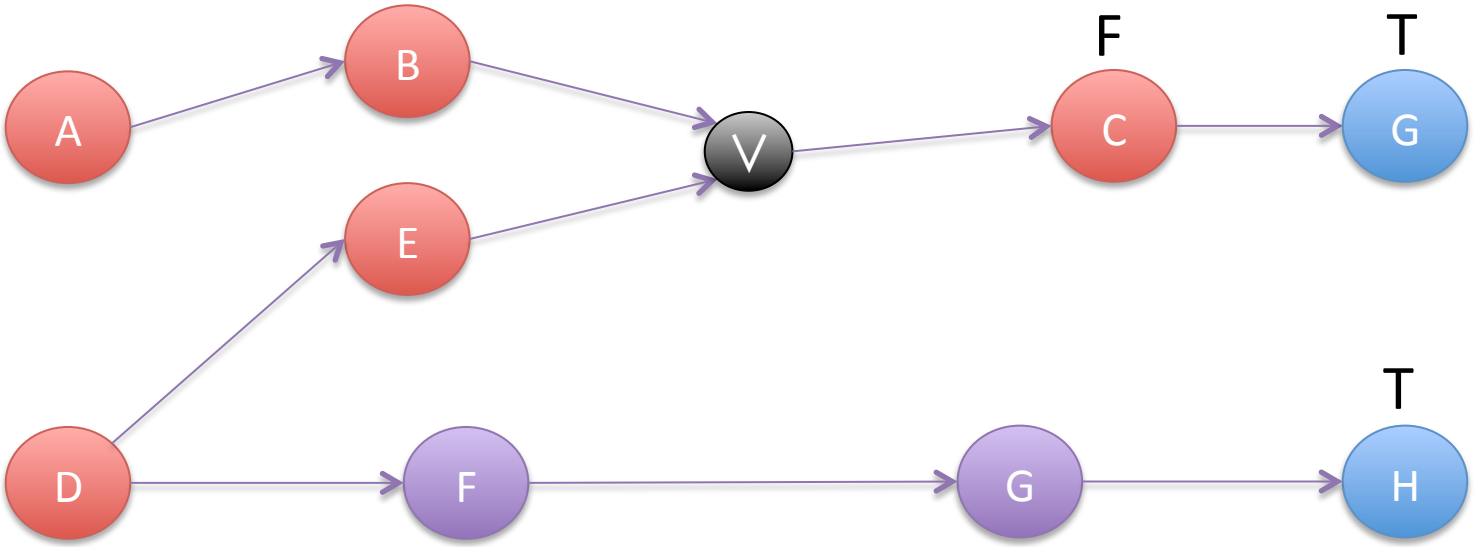
POSET (Partially Ordered Set)

- With contexts, form a poset and evaluate from the right to left until false



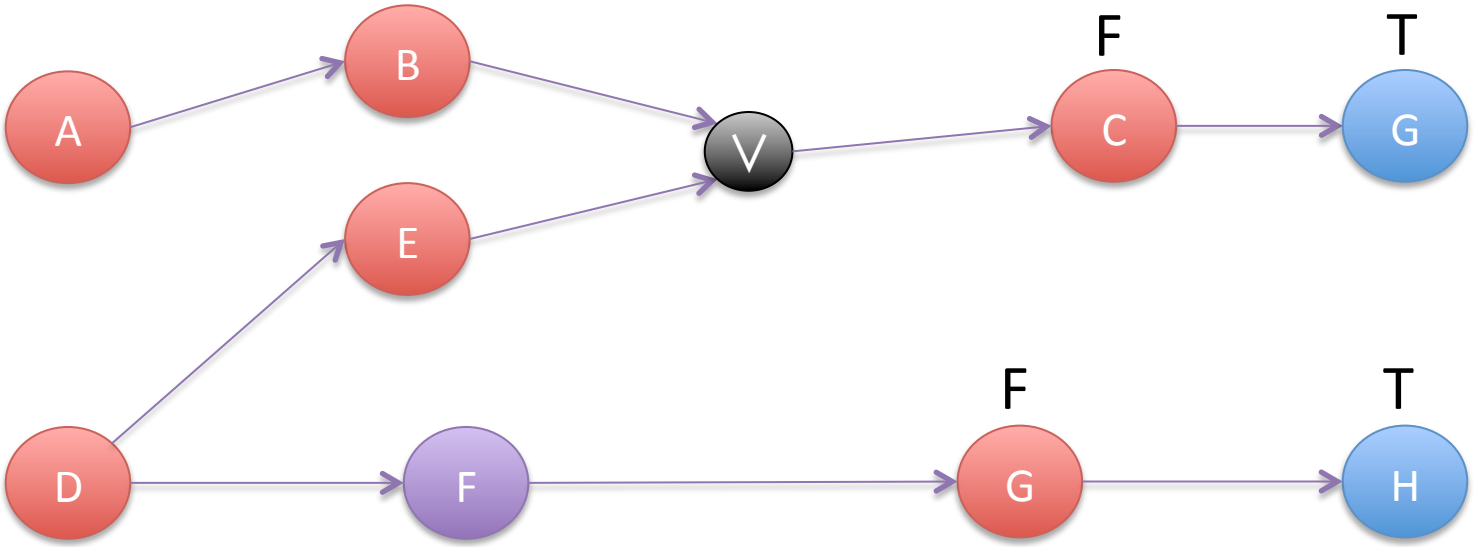
POSET (Partially Ordered Set)

- With contexts, form a poset and evaluate from the right to left until false



POSET (Partially Ordered Set)

- With contexts, form a poset and evaluate from the right to left until false



POSET (Partially Ordered Set)

- With contexts, form a poset and evaluate from the right to left until false