

User Modeling and Personalization

3: User Modeling - Techniques

Eelco Herder

L3S Research Center / Leibniz University of Hanover
Hannover, Germany

5 May 2014

Outline I

Reminder from past lectures

Overlay User Modeling

Introduction

The Domain Model

The Overlay Model

Propagation

In more detail: AHA!

System overview

Concepts and propagation

Knowledge propagation and authoring

Reasoning under uncertainty: Bayesian Networks

Introduction

Probability Theory

Outline II

Bayes' Theorem

Bayesian Networks

Calculating probabilities

Constructing a Bayesian Network

How to construct Conditional Probability Tables?

Diagnostic and predictive reasoning

Definitions from past lectures

User Model

A User Model is a data structure that characterizes a user U at a certain moment in time.

User data

User data consists of events and observations on the user's interaction with the system that can either directly be used for adaptation of that need to be resolved to user characteristics.

User Modeling

User Modeling is the process of creating and updating a user model, by deriving user characteristics from user data - which is either data that is explicitly provided by the user or data that stems from indirect events and observations.

Inference of knowledge

Knowledge inference is the process of interpreting events and observations on a user U , making use of conditions, rules or other forms of reasoning, and the storage of the inferred knowledge in the user model.

Hypertext

A Hypertext is a *graph* with uni- or bi-directional *edges*. A *node* represents a text document. An edge between two nodes represents a *link* between two text documents.

Adaptive Hypermedia Systems

“By adaptive hypermedia systems we mean all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user.”

User Modeling Techniques

In the previous lecture we distinguished the following (explicit) user modeling structures:

- ▶ *Flat model*: a simple collection of variables and associated values
- ▶ *Hierarchical model*: represents user characteristics and relations between these characteristics
- ▶ *Stereotype models*: Contains one or more stereotypes that are activated by triggers
- ▶ *Domain Overlay*: For each item in a domain, attributes represent the user's knowledge of or interest in this item
- ▶ *Logic-Based*: Representation and reasoning with first-order predicate logic

Stereotype user modeling is one of the oldest approaches to user modeling. Stereotypes were extensively used in early adaptive systems (between 1989 and 1994).

Overlay Modeling is the most important and most popular explicit user modeling approach in adaptive hypermedia.

Overlay User Modeling

Overlay User Modeling

An overlay user model represents an individual user's knowledge, interests, goals or other features as a subset of the domain model that resembles expert knowledge of the subject.

The original application of the overlay approach is *knowledge modeling*. As a natural extension, *overlay user interest* models have become popular as well.

Domain Model

The domain model decomposes the body of knowledge about a domain into a set of concepts (also known as knowledge items or topics).

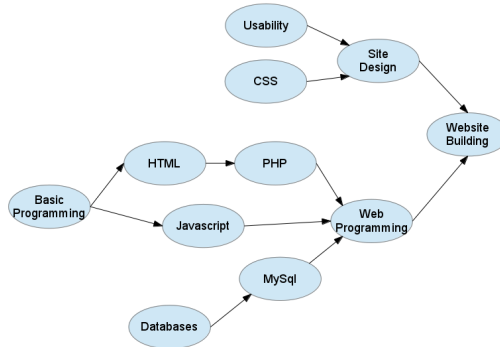
Flat models are the simplest form of domain models. Due to the lack of connections between concepts, the model is of limited use

Network models represent concepts and relationships between these concepts, thus forming an arbitrarily complex network.

Inspired by the Semantic Web, ad-hoc network models are replaced by formal *Ontologies*

Instructional Planning Models are a specific type of network models. It is formed by a tree of educational objectives to be acquired in a particular order. This model is popular in Learning Management Systems.

- ▶ in adaptive educational hypermedia systems the most popular relationship is 'prerequisite links'
- ▶ current systems also use classical *semantic* links, such as 'is-a' and 'part-of'
- ▶ relationships allow for the *propagation* of user characteristics to neighbouring concepts



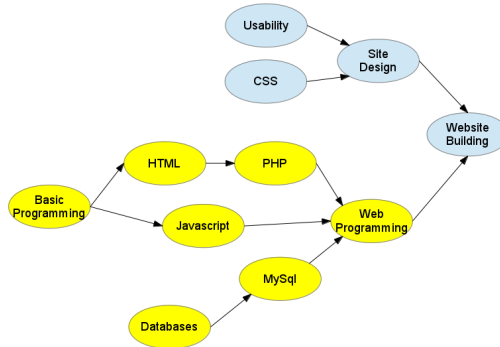
A simple domain model on skills required for web site design.
Arrows can be regarded as either as relationships or as prerequisite links.

The Overlay Model

The overlay model consists of (a subset of) the concepts from the underlying domain model. For each concept, the overlay model contains data that represents (an estimation of) the individual user's knowledge about or interest in this concept (or some other relationship with this concept).

Multiple overlay models can be attached to a domain model. Each model represents a different aspect of the users' relationship with the concepts.

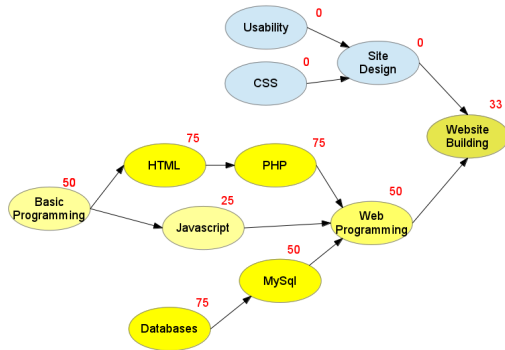
The simplest overlay knowledge model is the *binary overlay model*, with binary values 'known' and 'not known'.



A binary overlay model. The user knows the concepts that are marked yellow.

An extension is the *weighted overlay model* that can distinguish several levels of user's knowledge:

- ▶ *qualitative*: e.g. good - average - poor
allows for easy update and use of rules, used by rule-based systems
- ▶ *numeric*: quantitative values in some range (for example from 0 to 100)
used by systems with simple algebraic approaches to user modeling and propagation of characteristics
- ▶ *uncertainty-based*: probabilistic models, such as Bayesian networks, are used to model user characteristics explained in more detail in the second half of this lecture



A numeric overlay model.

Generalized overlay models may represent user features beyond knowledge and interests. In this case, the domain model may be a collection of goals, learning styles or stereotypes.

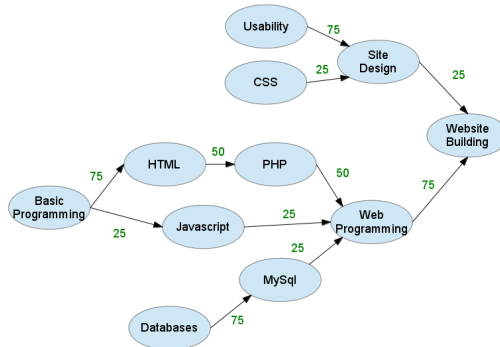
Given a Domain Model with prerequisite relations and a Knowledge Overlay Model. When the user demonstrates the presence of knowledge on an item, prerequisite links between concepts allow the propagation of the presence of knowledge beyond direct observation.

Different kinds of links may cause different types of propagation.

A very simple form of propagation was introduced in the Simple system of the first lecture:

“A document D_j is assumed to be learned by a user, if it has been visited or if a document D_k , for which D_j is a prerequisite, has been visited:”

$$\begin{aligned} &\forall U_i \forall D_j \\ &(\exists D_k \text{ preq}(D_k, D_j) \wedge \text{obs}(D_k, U_i, \text{Visited})) \\ &\implies \text{p_obs}(D_j, U_i, \text{Learned}). \end{aligned}$$






A simple knowledge propagation model. With basic programming knowledge, 75% of HTML knowledge is covered (with 25% still to learn), but only 25% of Javascript knowledge (with 75% still to learn).


In more detail: AHA!


AHA! is a simple Web-based adaptive hypermedia system, developed by Paul De Bra at the University of Eindhoven, the Netherlands.

[Home](#) 
[Adaptive Course Text](#) 


[Adaptive Course](#) 
[Text](#) 


[Assignments](#) 

[Announcements](#) 

[Roster](#) 

[Site Info](#) 

[Messages](#) 

[Help](#) 

Hypermedia Structures and Systems

Welcome back to the hypermedia course at the Eindhoven University of Technology.

This course contains the following (not necessarily disjoint) parts:

- [Introduction](#) (it is advised to read this before the other items)
- [Definition of hypertext and hypermedia](#)
- [The history](#) of hypertext and hypermedia

We advise to first study the above chapters, and then proceed with the more advanced chapters below.

- The architecture of hypertext systems
- Navigation (and browsing semantics) in hypertext
- Information Retrieval using hypertext
- Creating (or authoring) hypertext
- Multimedia aspects of hypermedia
- Open hypermedia and Web-Based Information Systems
- Distribution and Concurrency issues
- Adaptable and Adaptive Hypermedia
- The Future of Hypertext and Hypermedia
- Assignment for this course. This item only becomes available when you have finished most of the other sections.



An application consists of a number of *concepts*. Some concepts represent Web-page,s while others represent higher-level or abstract entities.

The *user model* (for each user) consists of a value for each concept. The most common interpretation of this value is that the value means the knowledge level of the user.

Each time a user visits a page, the value corresponding to that page is increased.

In AHA! the author can define *propagation rules* that cause this increase to influence the value for other concepts.

It is thus easy to define a structure of pages, sections and chapters:

- ▶ reading a page contributes to the knowledge of a section
- ▶ a knowledge increase for a section implies a (smaller) knowledge increase in the chapter.

The propagated changes are arbitrary. It is also possible to register a decrease in knowledge of a concept by accessing a certain page.

The author can define *requirements* for fragments of a page and for whole pages. Requirements are Boolean expressions over knowledge values for concepts.

When the requirement for a fragment of the page to be presented is true, the fragment is included in the presentation. (Otherwise the fragment is left out.)

When the requirement for a page is true, all anchors of links to this page are shown in a clearly visible color (typically blue or purple in AHA!).

Otherwise the anchors are shown in black, meaning that they will be hidden in the text. (The black links are also not underlined to make them effectively hidden.)

Concepts in AHA!

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE conceptList SYSTEM 'concept.dtd'>
<conceptList>
<name>Belgian Beer Example</name>
<concept>
  <name>de_koninck</name>
  <desc>first author's favorite beer</desc>
  <resource>de_koninck.xhtml</resource>
  <req>beer.interest > 50</req>
  <attribute name="access" type="bool"
    isPersistent="false" isSystem="true"
    isChangeable="false">
    <default>false</default>
    1 <generateListItem isPropagating="true">
      <req>beer.interest < 100</req>
      <trueActions>
        <action>
          <concept>beer</concept>
          <attribute>interest</attribute>
          <expr>beer.interest + 10</expr>
        </action>
      </trueActions>
    </generateListItem>
    2 <generateListItem isPropagating="true">
      <req>chocolate.interest < 50 and
        chocolate.interest > 4</req>
      <trueActions>
        <action>
          <concept>chocolate</concept>
          <attribute>interest</attribute>
          <expr>chocolate.interest-5</expr>
        </action>
      </trueActions>
    </generateListItem>
    3 <generateListItem isPropagating="true">
      <req>beer.interest > 50</req>
      <trueActions>
        <action>
          <concept>de_koninck</concept>
          <attribute>knowledge</attribute>
          <expr>100</expr>
        </action>
      </trueActions>
    </generateListItem>
    <attribute name="knowledge" type="bool"
      isPersistent="true" isSystem="false"
      isChangeable="true">
      <default>0</default>
      <generateListItem isPropagating="true">
        <req>true</req>
        <trueActions>
          <action>
            <concept>beer</concept>
            <attribute>knowledge</attribute>
            <expr>beer.knowledge + 0.2 *
              de_koninck.knowledge</expr>
          </action>
        </trueActions>
      </generateListItem>
    </attribute>
  </concept>
</conceptList>
  
```

Figure: AHA! Domain Concept: De_Koninck (Belgian Beer)

The concept (and its related page) is considered *desirable* by the system if the requirement *beer.interest* > 50 is fulfilled.

When a user accesses the page the attribute “access” temporarily becomes true. The event causes the three actions associated with this attribute to be considered for execution.

- ▶ The first action increases the *interest in beer*, but only if this interest is not yet 100 or more.
- ▶ The second action decreases the *interest in chocolate*, but only if this interest is not yet known to be high (but high enough to not become negative after the decrease).
- ▶ The third action registers that for interested users the knowledge about De Koninck becomes 100 and for uninterested users (who happen to find the page even though the links to it are hidden) the knowledge becomes only 35.

As one can see from the example, accessing the De Koninck page changes the interest in beer and in chocolate and raises the knowledge of De Koninck; this knowledge change triggers the rule for the knowledge attribute of De Koninck, which will then raise the knowledge of beer.

The change in interest for beer and chocolate and the change in knowledge of beer may also trigger rules associated with these concepts and attributes but that cannot be seen from the example as the concept structure of beer and chocolate is not given.

Knowledge Propagation

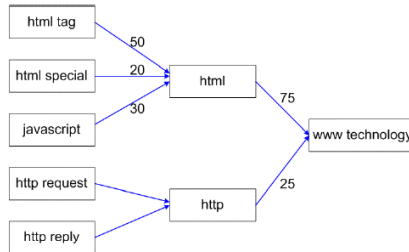


Figure 1: knowledge (update) structure in a course.

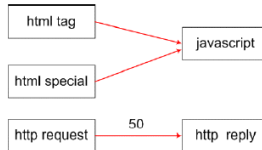


Figure 2: prerequisite relationships in a course.

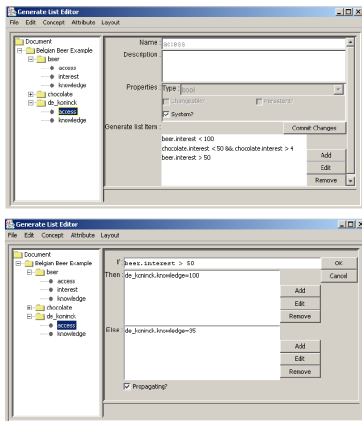


Figure: AHA! Authoring a Domain Concept

User Model

```
<!DOCTYPE profile SYSTEM 'profile.dtd'>
<profile>
  <record>
    <key>personal.id</key>
    <type>string</type>
    <persistent>true</persistent>
    <value>debra</value>
  </record>
  <record>
    <key>personal.email</key>
    <type>string</type>
    <persistent>true</persistent>
    <value>debra@win.tue.nl</value>
  </record>
  <record>
    <key>personal.goodlink</key>
    <type>string</type>
    <persistent>true</persistent>
    <value>0000ff</value>
  </record>
  <record>
    <key>personal.badlink</key>
    <type>string</type>
    <persistent>true</persistent>
    <value>000000</value>
  </record>
  ...
  <record>
    <key>de_koninck.knowledge</key>
    <type>int</type>
    <persistent>true</persistent>
    <value>35</value>
  </record>
  <record>
    <key>beer.knowledge</key>
    <type>int</type>
    <persistent>true</persistent>
    <value>20</value>
  </record>
  <record>
    <key>beer.interest</key>
    <type>int</type>
    <persistent>true</persistent>
    <value>40</value>
  </record>
  <record>
    <key>chocolate.knowledge</key>
    <type>int</type>
    <persistent>true</persistent>
    <value>15</value>
  </record>
  <record>
    <key>chocolate.interest</key>
    <type>int</type>
    <persistent>true</persistent>
    <value>20</value>
  </record>
  ...
</profile>
```

Figure: AHA! Fragment of a user model

Reasoning under uncertainty: Bayesian Networks

In user modeling, there is often the need to deal with information that is

- ▶ uncertain: we are not sure that the available information is absolutely true
- ▶ imprecise: the values handled are not completely defined

Examples of uncertain and imprecise information

- ▶ the user failed this question, so *most probably* he/she doesn't know concept C (uncertain information)
- ▶ the user has been reading about concept C for *quite* a long time (imprecise observation)
- ▶ the user is above 50, female and mother, so it is *likely* that she is a Book Reader (uncertain stereotype)
- ▶ the user passed a test on www technology, so we assume that she knows about hypermedia already (imprecise propagation)

Many observations on the user are not that easily interpretable.

Did a user read and understand all Web pages that he visited?

We could use some heuristics and assumptions in our rules:

- ▶ Ignore all page visits that took less than two seconds (user immediately clicked away)
- ▶ Assign *little knowledge/interest* to all pages that have been visited between 2 and 10 seconds
- ▶ Assign *medium knowledge/interest* to all pages visited between 10 seconds and 1 minute
- ▶ Assign *high knowledge/interest* to all pages visited between 1 and 5 minutes
- ▶ Ignore all pages visits that took longer than five minutes (coffee break?)

How correct are these assumptions? Can we assign 100% probability to them or just 80%?

If it is true that the user is interested in the page's theme, would the user be interested in similar themes? Are we sure?

Therefore, it is important to deal with uncertainty in user modeling

Some elementary definitions on Probability Theory

The *unconditional* (or *prior*) probability $P(A)$ is the degree of belief that a certain proposition holds - in the absence of any other information. For example:

$$P(\text{Rain} = \text{true}) = 0.4 \text{ or } P(R) = 0.4$$

All probabilities are values between 0 and 1: $0 \leq P(A) \leq 1$

Necessarily true (i.e., valid) propositions have probability 1,
necessarily false (i.e., unsatisfiable) propositions have probability 0:
 $P(true) = 1, P(false) = 0$

The probability of a disjunction is given by

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

$P(A)$ contains all cases where A holds, $P(B)$ contains all cases where B holds. Both sets contain cases where both A and B hold, so these need to be subtracted.

From the above follows that $P(Rain) + P(\neg Rain) = 1$ (it is necessarily true that it either rains or does not rain) and that $P(\neg Rain) = 1 - P(Rain) = 0.6$

Probabilities are pairwise independent.

$$P(Rain \cap Windy) := P(Rain) * P(Windy)$$

Conditional probabilities

Conditional probabilities indicate the probability that a proposition holds, given evidence for another proposition.

$$P(A|B), \text{ for example } P(Rain|Cloudy)$$

Conditional probabilities can be defined in terms of unconditional probabilities:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

which can also be expressed as

$$P(A \cap B) = P(A|B)P(B) \text{ (product rule)}$$

Bayes' Theorem

“Bayes' Theorem:” (follows directly from the definition of conditional probabilities):

$$P(A|B) = \frac{P(B|A)}{P(B)} * P(A)$$

As an example, if we know that $P(Rain) = 0.4$ and $P(Cloudy) = 0.7$, and that it is always cloudy when it rains, we can calculate the probability that it will rain when it is cloudy:

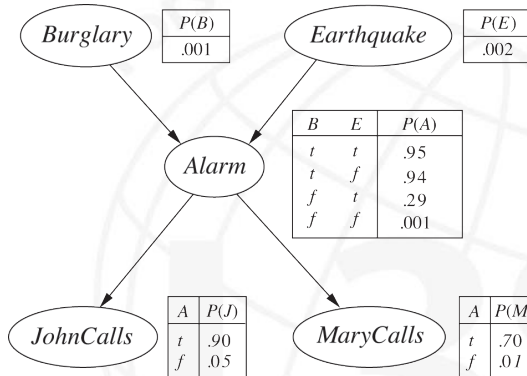
$$P(Rain|Cloudy) = \frac{P(Cloudy|Rain)}{P(Cloudy)} * P(Rain) = \frac{1}{0.7} * 0.4 = 0.57$$

Bayesian Network

A Bayesian network is a directed, acyclic graph with the following properties:

- ▶ Each node represents a random variable
- ▶ A (directed) edge from node X and Y indicates that X has a direct influence on Y
- ▶ The parents of a node are all those nodes that have arrows pointing to it
- ▶ Each node has a Conditional Probability Table (CPT) that quantifies the effects that the parents have on the node.

Example network: Burglary



Burglary Network, taken from Russel and Norvig, Artificial Intelligence

- ▶ A Bayesian Network represents a joint probability distribution
- ▶ Each entry in the joint probability distribution can be calculated from the information in the network.

$$P(X_1 = x_1, X_2 = x_2, \dots, X_n = x_n) = \prod_{i=1}^n P(X_i = x_i | \text{Parents}(X_i)) \quad (1)$$

The formula on the previous slide is a generalization of the definition of conditional probabilities:

$$\begin{aligned}P(A \wedge B) &= P(A|B) * P(B) \\P(X_1 \wedge X_2 \wedge \dots X_n) &= P(X_1|X_2 \dots X_n) * P(X_2 \dots X_n) \\&= P(X_1|X_2 \dots X_n) * P(X_2|X_3 \dots X_n) * P(X_3 \dots X_n) \\&= \dots \\&= \prod_{i=1}^n P(X_i|X_{i+1} \dots X_n)\end{aligned}$$

This can be simplified to equation 1 provided that each node is conditionally independent of its predecessors, given its parents.

Calculating probabilities

For the calculation of the conditional probabilities one only needs to consider the direct parents of each node.

For example, we can calculate the probability of the event that the alarm has sounded, but neither a burglary nor an earthquake has occurred, and both John and Mary call:

$$\begin{aligned} P(\text{John_calls}, \text{Mary_calls}, \text{Alarm}, \text{no Burglary}, \text{no Earthquake}) &= \\ &P(j, m, a, \neg b, \neg e) = \\ &P(j|a) * P(m|a) * P(a|\neg b, \neg e) * P(\neg b) * P(\neg e) \\ &= 0,9 * 0,7 * 0,001 * 0,999 * 0,998 \\ &= 0,00062 \end{aligned}$$

Or we can calculate the probability that John will call when there is a burglary, but no earthquake.

In this case, we can ignore the probability that Mary will call (that does not matter). We know that there has been a burglary and no earthquake, so these propositions are necessarily true ($P(b) = P(\neg e) = 1$). However, the alarm may go off or not (that has not been given).

$$\begin{aligned}
 P(j, a, b, \neg e) + P(j, \neg a, b, \neg e) &= \\
 P(j|a) * P(a|b, \neg e) * P(b) * P(\neg e) &+ P(j|\neg a) * P(\neg a|b, \neg e) * \\
 P(b) * P(\neg e) &= \\
 0,9 * 0,94 * 1 * 1 + 0,05 * 0,06 * 1 * 1 &= \\
 0,846 * 0,003 &= 0,849
 \end{aligned}$$

What is the probability that the alarm will go off when there is a burglary? This time we need to consider the cases when there is an earthquake, and when not.

Note that, for simplicity, we omit $P(b)$, as this is necessarily true. We also ignore whether John or Mary will call (we are only interested in the alarm).

$$P(a|b, \neg e) * P(\neg e) + P(a|b, e) * P(e) = \\ 0,94 * 0,998 + 0,95 * 0,002 = 0.938 + 0.002 = 0.94$$

From the above examples it can be observed that only the *Parent* nodes (and the parents of the parents) are needed to calculate the probability of an event (unless evidence is available from a child of the node - see next slide).

However, if the probability of a parent node has already been given (e.g. $P(a) = 1$), we do not need to consider whether a burglary or earthquake has taken place.

On the previous slide, we calculated the probability that the alarm will go off when there is a burglary. But we can also do this the other way round:

What is the probability that there has been a burglary when the alarm goes off?

For this, we need to make use of Bayes' Rule.

$$P(b|a) = \frac{P(a|b)}{P(a)} * P(b) =$$

So far so good. But in order to calculate $P(a|b)$ we need to consider both cases, when there has been an earthquake and not. And for $P(a)$ we need to consider all four possible cases when the alarm goes off (burglary or not, earthquake or not).

$$\frac{P(a, \neg e|b) + P(a, e|b)}{P(a, b, e) + P(a, b, \neg e) + P(a, \neg b, e) + P(a, \neg b, \neg e)} * P(b) =$$

This requires some calculation, but this equation is solvable.

$$\frac{0,94 \cdot 0,998 + 0,95 \cdot 0,002}{0,95 \cdot 0,001 \cdot 0,002 + 0,94 \cdot 0,001 \cdot 0,998 + 0,29 \cdot 0,999 \cdot 0,002 + 0,001 \cdot 0,999 \cdot 0,998} \cdot 0,001 =$$

$$\frac{0,019 + 0,938}{0 + 0,00094 + 0,00058 + 0,001} \cdot 0,001 =$$

$$\frac{0,9399}{0,00252} \cdot 0,001 = 0.37$$

Constructing a Bayesian Network

To guarantee that that a Bayesian network does not violate the axioms of probability, the semi-algorithmic construction of a Bayesian is as follows:

- ▶ Choose the set of relevant variables X_i that describe the domain
- ▶ Choose an ordering for the variables
- ▶ While there are variables left:
 - ▶ Pick a variable X_i and add a node to the network
 - ▶ Set $Parents(X_i)$ to some minimal set of nodes such that the conditional independence property is satisfied
 - ▶ Define the conditional probability table for X_i

How to construct Conditional Probability Tables?

Conditional Probability Tables are an essential part of the description of a Bayesian Network.

It is assumed that conditional probabilities are easier to estimate than (normal) probabilities. Compare:

- ▶ How likely is it that it will be raining tomorrow?
- ▶ How likely is it that the ground will be wet after a shower?

A common approach is to rely on *expert knowledge* for estimating the conditional probabilities.

An alternative approach is to automatically construct a Bayesian network (both CPTs and the relations) using (supervised) learning approaches. This approach is beyond the scope of this lecture series.

Diagnostic and predictive reasoning

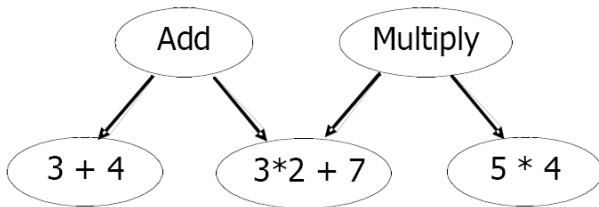


Figure: A basic BN for diagnosing knowledge about adding and multiplying

The Bayesian Network on the previous slide allows for two types of inference:

- ▶ *Diagnostic inference*: what are the more probable causes given certain evidences
e.g. compute the probability that the student knows the concept “Add” given the answer to $3 + 4$
- ▶ *Predictive inference*: what is the probability that a given configuration of variables will happen, given a set of evidence
e.g. compute the probability that the student will be able to correctly solve $3 * 2 + 7$

The same model allows us to estimate the user (knowledge) characteristics as well as to estimate future activities/results in order to provide sensible recommendations.