Reasoning With Uncertainty

Text Chapter 5
and 9: 9.2 only, to the end of 9.2.2

Logical Implication

- “All traditional logic habitually assumes that precise symbols are being employed. It is therefore not applicable to this terrestrial life but only to an imagined celestial existence.” - Bertrand Russell
- “So far as the laws of mathematics refer to reality they are not certain. And so far as they are certain, they do not refer to reality.” - Albert Einstein
- "In this world, nothing is certain but death and taxes." - Benjamin Franklin

Logical Implication

- First order logic gives us the ability to derive new (not explicitly represented) axioms using logical inferences
- Everything we derive in FOL stems from the idea of logical inference – that we prove theorems that are a logical consequence of the axioms in the knowledge base

$$A \rightarrow A \rightarrow A \rightarrow A \rightarrow \text{Theorem}$$

(the arrow is logical implication)

- The truth of this theorem rests solidly on the mathematical certainty of logical inference

Logical Implication

- In order to be able to use implication to generate conclusions our statements have to be axioms (completely true statements), and our logical system must be correct (no inconsistencies) and complete (no missing knowledge) in order to be able to use logical implication to generate conclusions
- If the knowledge base is incomplete, the system will not be able to prove theorems
- If the knowledge base is inconsistent, the conclusions will not necessarily be valid

The Real World

- Of course does not work like this!
- Most everything is uncertain to some degree
- So much so that the more you think about it the more you wonder why anything works at all
- Yet we deal with this relatively easily (unless there's massive amounts of uncertainty – even then, lotteries are still popular!)

Reasoning With Uncertain Knowledge

- We need to be able to reason logically *IN SPITE* of uncertainty in order to solve problems in the real world
- That is, we need to take uncertainty into account in our reasoning. Conclusions will necessarily not be guaranteed complete and correct (like the real world!)
- Would like to be able to generate conclusions that are correct most of the time with respect to the domain but which may not be logically implied by the axioms

Reasoning With Uncertain Knowledge

- To do this, the system must use additional supports when necessary to deal with uncertainty
- These supports can be used to fill in the gaps in uncertain knowledge – to take the uncertainty into account

$$A \rightarrow_{s} \rightarrow A \rightarrow_{s} \rightarrow \text{Conclusion}$$

additional supports for the reasoning process!

Reasoning With Uncertain Knowledge

- The result of the inference process is a “conclusion”, not a formal theorem
  - the conclusion is probably (hopefully very often!) true but is not *guaranteed* to be true
- No matter what the source of the uncertainty, this type of reasoning falls under the general heading of *Reasoning Under Uncertainty*
- However there are really two broad types of such reasoning, and we'll consider them separately

Reasoning with Incomplete Knowledge

- One of the most common problems encountered in real world situations involves dealing with incomplete knowledge
- In these situations, we have to draw conclusions in spite of the fact that we don't have access to all potential information that might influence that decision
- For example, we can represent the fact that birds can fly

  If bird(X) Then fly(X)

Reasoning with Incomplete Knowledge

- However, there are exceptions to this rule; for example, ostriches can not fly

  If bird(X) and ~ostrich(X) Then fly(X)

- There are additional exceptions, such as baby birds, birds with broken wings, dead birds (and many other examples that quickly stretch the bounds of good taste), etc.

  If bird(X) and ~ostrich(X) and ~baby(X) and ~dead(X) Then fly(X)

Reasoning with Incomplete Knowledge

- The number of such qualifications is essentially *endless* (there may be qualifications to the qualifications)
  - e.g. dead birds can't fly, unless you're in a cartoon
- Adding an infinite number qualifiers isn't possible
- And think of the amount of work we get into - we can't conclude that Fred can fly unless we can also prove that Fred is not an ostrich, is not a baby, is not dead, etc.

Reasoning with Incomplete Knowledge

- How far do we go proving each of these before it's acceptable? When do we stop? The process is essentially endless
- We need a significant amount of knowledge about the properties that Fred does not have in order to be able to make conclusions about the properties that Fred does have
- This is the Qualification Problem
- We have to add extra supports to FOL to be able to make inferences and avoid this

Closed World Assumption

- In most knowledge bases, there is a small number of positive facts compared with the total number of facts
- For example, an airline stores points that it provides flights between
  - not all the cities it doesn't!

    connect(96, winnipeg, toronto)

    connect(74, toronto, montreal)
- We do not represent negative information such as there is no direct flight from Winnipeg to Montreal or there is no flight from Winnipeg to Grand Beach

Closed World Assumption

- The closed world assumption states that if a fact is not represented explicitly in a knowledge base and the fact is not implied by the knowledge base, then the fact is assumed to be false
- The closed world assumption allows a system to make “reasonable” assumptions about knowledge that is missing from the system

Closed World Assumption

- So Fred would be assumed to be not an ostrich, not a baby, and not dead unless the system could prove otherwise
- The system must still expend effort to prove that each of the facts is not true before it can assume that they are false and then conclude that Fred can fly
- The closed-world assumption is referred to in Prolog as ***negation as failure***: failure to prove a statement is taken to mean that the statement is false

Negation as Failure

- A problem with negation as failure is differentiating between a statement that is false and a statement for which the truth can't be determined

```
? alive(tweety).
no.
? dead(tweety).
no.
?- not alive(tweety).
yes.
?- not dead(tweety).
yes.
```

Negation as Failure

- Unless we know ***explicitly*** that alive(tweety) is true or that dead(tweety) is true, the negation of alive(tweety) will be true and the negation of dead(tweety) will also be true
  - this leads to multiple interpretations (via backtracking) of the knowledge which is typically not acceptable
  - we have to eliminate the multiple interpretations by including axioms that allow us to resolve them
  - e.g. saying that not being alive implies dead
  - What is this really doing?

Reasoning with Defaults

- What is the problem with the following rule?

  If ostrich(X) Then not fly(X)
- Prolog does not permit rules that begin with a negative
- When combined with the rule that all birds can fly this leads to a contradiction (via backtracking)

  ostrich(fred)

  not fly(fred)

  fly(fred)
- While the intent of this rule may be correct, it does not provide the correct result

Reasoning with Defaults

- In the previous examples, there is no knowledge that indicates what knowledge should be assumed in the absence of other knowledge
- Defaults can be used to supply knowledge that is missing from an incomplete knowledge base
- The implementation of defaults normally requires the use of negation as failure

Reasoning with Defaults

- Defeasible Reasoning
- Default rules have the following form

  If (p(X) and ~ab(X)) Then g(X)
- This rule states that if p(X) is true and it can not be proved that X is abnormal (ab(X)), then conclude that g(X) is true
- If ab(X) does not exist as a fact or an inference, then the system can assume that g(X) is true

Reasoning with Defaults

- The fact that specific birds can not fly is represented explicitly

  If ostrich(X) Then not_fly(X)
- The system will now conclude that ostriches can not fly but that all other birds can fly

  If (bird(X) and ~ not_fly(X)) Then fly(X)

Reasoning with Defaults in Prolog

- Default knowledge is represented in Prolog in a similar manner

  bird(X) :- ostrich(X).

  not_fly(X) :- ostrich(X).

  fly(X) :- bird(X), not not_fly(X).
- The predicate not_fly(X) is the abnormal predicate
- Note that a positive term is included first in the fly predicate

Reasoning with Defaults in Prolog

- In the previous example, there is only one valid interpretation for a specific set of facts and rules
  - either a bird can fly or it can not fly
  - both interpretations are not possible
- The abnormal predicate (not_fly) blocks the use of the default rule (fly) when a bird is abnormal (i.e. can not fly)

Reasoning with Defaults in Prolog

- For the statements below, 2 interpretations are possible (fly and not_fly) for the same object because the default rule is not blocked
- Prolog will prove not_fly for an ostrich and then will prove fly via backtracking

  bird(X) :- ostrich(X).

  not_fly(X) :- ostrich(X).

  fly(X) :- bird(X).

Reasoning with Defaults in Prolog

- Suppose some penguins have been genetically altered so that they are able to fly

```
genetic_penguin(opus).
penguin(X):- genetic_penguin(X).
bird(X) :- penguin(X).
not not_fly(X):- genetic_penguin(X).
not_fly(X) :- penguin(X).
fly(X) :- bird(X), not not_fly(X).
```

- This is not valid Prolog

Reasoning with Defaults in Prolog

```
genetic_penguin(opus).
penguin(X):- genetic_penguin(X).
bird(X) :- penguin(X).
can_fly(X) :- genetic_penguin(X).
not_fly(X) :- penguin(X), not can_fly(X).
fly(X) :- bird(X), not not_fly(X).
```

- By structuring the rules in this manner, there is only one possible interpretation for each bird

Multiple Extensions

- Some problems cannot easily be defined so that they do not produce contradictions
- This type of reasoning involves multiple extensions
- Quakers are pacifists; Republicans are not pacifists; Nixon is a Quaker and a Republican (Ginsberg, p. 218)
- Is Nixon a pacifist or not?
- Unless one rule blocks the other rule, both interpretations are possible
- classic multiple inheritance problem!

Defeasible Reasoning

- A significant amount of care must go into the representation of reasoning with defaults in Prolog
- An extension of Prolog, d-Prolog, has been designed to provide the facilities required for default reasoning
- d-Prolog takes advantage of the facility in Prolog for defining new operators

Monotonic Knowledge

- Classical inference in first-order logic is monotonic
- Any sentence that is true in one knowledge base is also true in any knowledge base that is a superset of this knowledge base
- Adding new knowledge does not invalidate existing knowledge

Nonmonotonic Knowledge

- A knowledge base that uses negation as failure is nonmonotonic
- Adding a new piece of knowledge may invalidate a previous conclusion

```
bird(opus).
?-  fly(opus).
yes.
penguin(opus).
?-  fly(opus).
no.
genetic_penguin(opus).
?- fly(opus).
yes.
```


Nonmonotonic Knowledge

- A nonmonotonic knowledge base should still have unique interpretations
  - a specific object can either fly or it can not fly
  - both interpretations of the same knowledge should not be possible

Project Proposals

- FOCUS ON THE AI
  - Many areas have elements that are reliant on hardware or non-AI software
    - image capture, parsing,…
  - These are not interesting from an AI standpoint. Mentioning their existence is fine but you shouldn't be spending a lot of time on them
  - Ditto applications – again, this is NOT an application project
    - There were proposals that said "this is not just 'ai In X'", but then produced an outline that easily could be!

Working on your Projects

- There are numerous things you should be thinking about
- Depth
- Some component that is your argument / idea / implementation / experimentation… i.e. a contribution beyond papers you've read
- Reasonable sources for your area of choice
- i.e. published in a peer-reviewed journal/conference/book

Sources

- This excludes popular press articles (e.g. newspapers, many magazines)
- You want things that are addressing, for the most part, other people who study this area
- Does this mean I can't get it from the web?
  - No! Many people, myself included, make their publications available on the web. The point is that it IS published elsewhere, that elsewhere is reasonable, and you know where that elsewhere is
  - Most people will (should) state this information clearly. If they don't you should be suspicious

Sources

- Make sure you strongly differentiate sources that are ***made available via the web***, even though they're published elsewhere, and something that's ***just a website***.
- Websites themselves have no scientific reputation and should be avoided
- You need to cite ***every source every time*** you use it – that is how one differentiates one's own work from that of others!

For Example

There are many approaches to individual-based modelling, from creating individuals as cellular automata with simple transition and interaction rules ([Slothower et al., 1996]), to modelling individuals with broader perceptual/effectory abilities (e.g. Gecko [Booth, 1996] and the earlier SWARM system [Hiebler, 1994] on which it is based). However, computing resources and associated restraints on the complexity of individual models are still the largest limiting factors in individual-based models, along with a lack of suitable tools for developing such models. Many individual-based ecosystem models manage large numbers of individuals (e.g. [Wilson et al., 1993; Schmitz and Booth, 1997]) while maintaining only a very small amount of information on each individual or supporting a fairly limited variation on individual physiology and behaviour. The need to maintain more than a few parameters of interest for each individual is also recognized as one of the most significant factors in deciding whether individual-based modelling is feasible [Judson, 1994].

Sources

- Your bibliography then contains complete information about where these publications can be found

  [Schmitz and Booth, 1997] Schmitz, O. J., and G. Booth, 1997. "Modelling Food Web Complexity: The Consequence of Individual-Based Spatially Explicit Behavioural Ecology on Trophic Interactions", *Evolutionary Ecology* 11(4): 379-398.

- Your bibliography contains only information you specifically cited in the text you have written (i.e. every entry has at least one citation, and there is an entry in the bibliography for every citation)

Citations

- Again, it is vitally important that you cite properly – if you omit citing information the implicit claim is that *you* thought of this
  - which is academic dishonesty
- Direct quotes must be placed in direct quotes – taking large sections from another paper directly or changing the wording trivially is academic dishonesty
- There is no need to directly quote much, unless the precise wording is vital: direct quotes should be rare, *the writing should generally be in your own words*

Organization

- Your work should have an abstract at the top describing in a short paragraph what you're going to be doing (no more than 100-150 words)
- You should have an introduction, followed by a series of sections that introduce and expand your topic, and finally, conclusions
- Likely the best way to keep good organization if you have not written anything substantial before is to do your sections in point form (i.e. the main points you intend to make) and then expand these to full text)

Organization

- Sometimes what you write isn't particularly good the first time around, and if you write full text at the start you're more likely to be tempted to keep something around just because you put the effort into writing it
- Make sure you know what you want to say before you say it
- It is generally obvious to a reader (at least this one) when you're using real content and when you're attempting to fill space

Reasoning with Imprecise Knowledge

- It has been assumed that at any specific point in time each statement is either true or false - there can be no vagueness associated with a statement

  male(bill)

  parent(bill, sally)

- Not possible to be kind of male, or sort of a parent…it's a binary thing
- Most concepts, however, aren't like this!

Reasoning with Imprecise Knowledge

- Much of the knowledge that is used in real world situations is not *either* true or false; instead, it is true some of the time but not all of the time
- Such knowledge is referred to as imprecise knowledge

  If sore-throat(X) then have-cold(X)

- We could use default rules to block this conclusion in cases we know about
- More often we want to know how **likely** it is you have a cold given you have sore throat!

Representing Imprecise Knowledge

- To represent imprecise knowledge, we need to be able to represent the *degree of truth* of a statement
- We can represent this knowledge by assigning a numerical measure to each statement in some fashion
- We first need to decide what kind of numeric scale we're going to use…

Representing Imprecise Knowledge

- For example, a true statement could have a measure of +1, a false statement could have a measure of -1, and other statements would have floating point measures between -1 and +1
- A statement with a measure of 0.9 would be true most of the time, but not all of the time

Now that we have a scale

- We have two classic problems that come up ***whenever*** we reason with imprecise information….

First, Where do the numbers come from?

- Statistics?
- Estimates based on lots of experience (e.g. ask our expert)
- Average over lots of experts?
- Good general guesses?
- e.g. why 0.85, and not 0.84 or 0.86?

The Second Problem

- What do we DO with these numbers when we make inferences?
- e.g. if I have some imprecise measure of having a sore throat (i.e. I'm not certain I do, I think I might be getting one)
- And I have a rule with some other likelihood that that means I have strep throat
- Then what's the *overall* likelihood that I have strep throat?

Propagating and Combining Measures

- Or if I have another rule that says that if I have a sore throat, I should likely have some antibiotic, then what's the overall likelihood of having the antibiotic?
- Any scheme for dealing with imprecise reasoning must have some valid way of making inference after inference, propagating or combining the measures we put into each

Certainty Factors

- Certainty factors were developed for use in the Mycin expert system in order to represent and manipulate the numerical measures associated with imprecise statements
- A certainty factor is a measure of belief that a statement is true
- A certainty factor may be associated with a simple fact or with a rule

Certainty Factors

- For example, the belief that it will be cloudy tomorrow can be represented by including a certainty factor with a fact

  cloudy-tomorrow $_{0.5}$
- Similarly, the belief that if it is cloudy tomorrow it will also rain can be represented by including a certainty factor with a rule

  If (cloudy-tomorrow) Then rain-tomorrow $_{0.75}$

Manipulating Certainty Factors

- The certainty factors must be combined in such a way that the final certainty factors are reasonable, given the characteristics of the domain
  - cloudy-tomorrow $_{0.5}$
  - If (cloudy -tomorrow) Then rain-tomorrow $_{0.75}$

    so -

    rain-tomorrow$_{??}$

Manipulating Certainty Factors

- Even within the general realm of certainty factors there are numerous means of combining these – they arose as *ad hoc* approaches for particular expert systems
- There is no "best" technique for combining certainty factors - different techniques work well in different domains and poorly in others

A Typical Method

If (Ax) Then Cy

- The resulting CF of C is x*y – the certainty of the result of a rule is affected by the certainty of the facts used in that rule

If (Ax ^ By) Then ...

- The resulting CF of the premise is min(x,y)

If (Ax v By) Then ...

- The resulting CF of the premise is max(x,y)
- we're only as certain as the lowest participant in a conjunction, but certain to the maximum participant in a disjunction

Manipulating Certainty Factors

If (Ax) Then C

If (By) Then C

- Here we have 2 different rules being applied to give c
- Think of a medical domain, where I have a symptom and it indicates some disease
- A second symptom indicating that disease further (in fact usually much more) supports our certainty in C
- Mathematically:

Manipulating Certainty Factors

If (Ax) Then C

If (By) Then C

- If the evidence is independent, the resulting CF of the premise is

x+y-(x*y) if x and y > 0

- The resulting CF of the premise is

x+y+(x*y) if x and y < 0

- Otherwise, the resulting CF of the premise is

(x+y)/(1-min(|x|,|y|))

Manipulating Certainty Factors

If ($A_{0.5}$) Then C

If ($B_{0.75}$) Then C

The CF of C is = .5 + .75 - (.5 * .75) = 0.875

If ($A_{-0.5}$) Then C

If ($B_{-0.75}$) Then C

The CF of C is = -.5 + -.75 + (.5 * .75) =

-0.875

Manipulating Certainty Factors

If ($A_{-0.5}$) Then C

If ($B_{0.75}$) Then C

The CF of C is $= (-.5 + .75) / (1-0.5) = 0.5$

Certainty Factors

- A medical diagnosis system could make diagnoses based on the symptoms that are present

  If (left-arm-pain $\wedge$ neck-pain) then spinal-cord-injury

  If (shortness-breath $\vee$ chest-pain) then over-exertion

  If (chest-pain) then cardiac-arrest

  If (left-arm-pain) then cardiac-arrest

  If (left-arm-pain) then rotator-cuff-injury

Certainty Factors

- It is important to differentiate between symptoms that are independent and symptoms that are related

  If (left-arm-pain ^ neck-pain) then spinal-cord-injury

- ***Both*** symptoms must be present for the conclusion to be true – the rule is useless without both

  If (shortness-breath$\vee$chest-pain) then over-exertion

  – These two symptoms are related and do not increase the confidence in the diagnosis if both are present – if they were independent we would put them in separate rules, and they'd be combined using the formula that does increase confidence by both being present

Certainty Factors

If (chest-pain) then cardiac-arrest

If (left-arm-pain) then cardiac-arrest

- These two symptoms are independent: either symptom can lead to the conclusion and both symptoms together increase the confidence in the conclusion

Certainty Factors

left-arm-pain(george)$_{0.7}$
neck-pain(george)$_{0.8}$

if (left-arm-pain(X)^neck-pain(X)) then
spinal-cord-injury(X)$_{0.9}$

spinal-cord-injury(george)$_{0.7*0.9=.63}$

Certainty Factors

chest-pain(george)$_{0.8}$

if(shortness-breath(X) v chest-pain(X)) then
over-exertion(X)$_{0.9}$

chest-pain(george)$_{0.8}$
over-exertion(george)$_{0.9*0.8=.72}$
but if we had:
shortness-breath(george)$_{0.7}$
over-exertion(george)$_{0.9*0.7=.63}$

think about backward reasoning with this – we now have to explore both branches, to get an accurate picture of certainty!

Certainty Factors

if (chest-pain(X)) then cardiac-arrest(X)$_{0.9}$
if (left-arm-pain(X)) then cardiac-arrest(X)$_{0.9}$

chest-pain(george)$_{0.7}$
gives us cardiac-arrest(george)$_{0.9*0.7=.63}$
left-arm-pain(george)$_{0.8}$
gives us cardiac-arrest(george)$_{0.9*0.8=.72}$
combining these:
cardiac-arrest$_{0.9*(0.7+0.8-(0.7*0.8))=.846}$

Certainty Factors

- For the following symptoms:
  left-arm-pain(george)$_{0.7}$
  neck-pain(george)$_{0.8}$
  shortness-breath(george)$_{0.7}$
  chest-pain(george)$_{0.8}$
- The following conclusions (diagnoses) would be made:
  spinal-cord-injury(george)$_{0.9*0.7=.63}$
  over-exertion(george)$_{0.9*0.8=.72}$
  cardiac-arrest(george)$_{0.9*(0.7+0.8-(0.7*0.8))=.846}$

Certainty Factors

- For this particular collection of symptoms, the most likely diagnosis for george is cardiac-arrest
- If a system based on logical implication had been used instead, all 3 diagnoses would have been generated but there would not have been an information that indicated which diagnosis was most likely
- The certainties associated with the individual symptoms were obviously not realistic - they were used solely to illustrate the process

Prolog Certainty Factors

- Implementing certainty factors in Prolog requires a change to the predicates used to represent knowledge plus the addition of predicates to manipulate the certainty factors

  symptom(fact).

  symptom(neck-pain).
- becomes

  symptom(fact, cf).

  symptom(neck-pain, 0.95).
- Prolog rules need to be written to modify the certainties of conclusions…

Certainty Factors

- Certainty factors permit the definition and manipulation of knowledge that is more complex than default knowledge
- Evidence can be combined from a variety of sources, with each piece of evidence contributing towards increasing or decreasing the confidence in the various conclusions
  - conflicting information can be processed

Certainty Factors

- Recall our two basic problems in imprecise reasoning
- In CF, like other approaches, the measures may be obtained from a statistical sampling or the measures may just be an estimate obtained from an expert in the domain
- The manipulation mechanisms are somewhat “ad hoc” -- they work but do not have any really strong theoretical basis

Certainty Factors vs Probabilities

- Certainty factors are different from probabilities
  - a probability of 1 means true
  - a probability of 0 means false
- A probability of 0.5 means that an event has a 50% chance of occurring
- A certainty factor of 0 means that we do not know if the event will occur or not
- The major advantage of certainty factors over probabilities is that combined probabilities decrease towards 0 (false) while certainty factors decrease towards 0 (not known)

Probabilistic Approaches

- The advantage to more probabilistic approaches is that the combination mechanisms have a much more formal basis
- i.e. what we're saying has a strong theoretical foundation, it's not just "kind of nice"
- There are numerous probabilistic approaches, many based on Bayesian Reasoning

Probabilistic Approaches

- the degree of belief in any statement is based on the evidence we accumulate for that belief
- This implies that evidence changes over time
  - and allows us to define two kinds of probablities
- Suppose I pick a card out of a deck
  - Before looking at the card, likelihood of a particular card is 1/52
  - After? 0 or 1

Prior vs. Conditional Probabilities

- We call an estimate of probability before any evidence is obtained a Prior probability
- An estimate after a particular piece of evidence is obtained is a Posterior or Conditional probability
  - because it's conditional on having that particular piece of evidence
- e.g. likelihood of patient having a cold in general vs. if they're coughing

Prior Probability

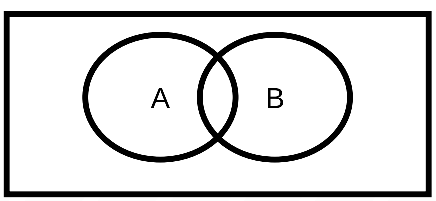
- probability of the proposition being true given no particular evidence
- e.g. $P(Cavity)=0.1$
- in the absence of any other information, 10% chance of someone having a cavity
- No longer prior probability as soon as we have any evidence at all

Conditional Probability

- Probability of cavity given we know someone has a toothache
- $P(cavity|toothache) = 0.85$
- If no other information other than the fact they have a toothache is available, we have a likelihood of a cavity of 85%
- the condition is important. If we know B and C are true, $P(cavity|B)$ is not useful - we must compute $P(cavity|B \wedge C)$

Axioms of Probability

- We need to describe how probabilities and logical connectives interact, and so define a few basic truths about probability:
  - $0<P(X)<1$: all probabilities between 0 and 1
  - $P(True) = 1$, $P(False) = 0$ (end point meaning)
  - $P(A \vee B) = P(A)+P(B)-P(A \wedge B)$ (look familiar???)



Defining Conditional Probability

- can also define conditional probabilities in terms of unconditional ones
- $P(A|B) = P(A \wedge B) / P(B)$
  - as long as $P(B)>0$
- Also written as $P(A \wedge B)=P(A|B)\ P(B)$
  - which makes more sense semantically to read!
  - We need B to be true, and then A to be true given B
  - also $P(A \wedge B) = P(B|A)\ P(A)$
- This is the *Product Rule*

Joint Probability Distribution

- A Joint Probability Distribution completely assigns probability assignments to all propositions in the domain
- The joint is thus an N-dimensional table with a value in every cell stating the probability of that specific state occurring
- Consider an example: toothache vs. cavity…

Joint Probability Distribution

	Toothache	~Toothache
Cavity	0.04	0.06
~Cavity	0.01	0.89

- the atomic events are mutually exclusive, so any conjunction has to be false
- therefore, the entries must sum to 1 (by the 2^{nd} & 3^{rd} axioms of probability)

Using the Joint

- We can use the joint to compute any probabilistic statement we want about the domain
- simply by stating the statement in terms of a disjunction of joint entries, and adding up the respective probabilities
- Adding a row or col gives us unconditional probabilities:
  - P(Cavity) = 0.06+0.04 =0.10
  - P(Cavity V Toothache) = 0.04+0.01+0.06=0.11
    - note that a cell is only added in once!

Calculating conditional probabilities?

- We can use the product rule and the joint to calculate conditional probabilities given the collection of unconditional probabilities we have here
- For example, P(Cavity|Toothache)?
  - P(A|B) = P(A^B) / P(B)

  = P(Cavity ^ Toothache) / P(Toothache)

  = 0.04/(0.04+0.01) - read off from the joint

  = 0.04/0.05

  =0.8
- What's the problem with doing this?

Problem

- There may be thousands of random variables in a given problem – easy for 2 (toothache and cavity), but gets difficult quickly as we increase
- Can be a nasty, nasty number of combinations!
- So we need something to come to our rescue: Bayes' Rule

Bayesian Reasoning

- Bayes' rule is derived from the two forms of the product rule we used earlier. Equating them and simplifying, we get
- P(B|A) = (P(A|B) P(B)) / P(A)
- This is Bayes' rule, which forms the foundation for most modern probabilistic reasoning systems in AI
- doesn't look particularly useful at all - we're using three terms just to calculate one CP
- Turns out in practice we DO often have good guesses for these three terms though!

Using Bayes' Rule

- Medical Diagnosis:
- A doctor knows meningitis causes a symptomatic stiff neck a particular portion of the time
- We can look at how often stiff necks occur in the general population, and the rate of occurrence of meningitis
- Bayes’ rule will allow us to calculate the likelihood of a meningitis diagnosis being the correct one if a patient exhibits a stiff neck

Using Bayes' Rule

- Want P(M|S)
- P(S|M) = 0.5 (observable from records)
- P(M) = 1/50000, P(S)=1/20 (observable statistically)
- P(M|S)=(P(S|M) P(M))/P(S)
- =(0.5*1/50000) / (1/20)
- =0.0002
- which is what you'd expect: really really really unlikely!
- How does this compare with using a textbook value that might be obtained for this?

Using Bayes' Rule

- It's a whole lot more flexible
- If you read that 1/5000 of people with stiff necks had meningitis, and there was an epidemic of meningitis at the time, not clear how this would affect your diagnosis
- Using Bayes' rule, P(M) causes the likelihood of our diagnosis being meningitis to rise appropriately
- This is a more direct causal model, recording what's going on in the environment

Downside of this

- requires symptoms to be independent to be useable in practice
- Calculations like we did a few moments ago are easy with Bayes' rule
- Doing things like accumulating evidence though, isn't
  - can be really nasty to do probabilistically because of the number of possibilities
- For accumulating evidence, we typically relax some of the independence constraints to make things computable, and only concentrate on the possible combinations we're interested in

Bayesian Belief Networks

- We effectively construct a graph of what facts influence what other facts
- When evidence increases our belief in some fact, we propagate this belief probabilistically to the connected facts
- The network gets altered and even rewired as new facts are added and old facts removed (see text section 9.3.1)

Imprecise Concepts

- Reasoning with uncertainty is further complicated by the idea that the concepts we talk about are also not precise in many cases
- For example, I might talk about somebody being tall, or having an awful headache, or being incredibly drunk
- How tall is tall? How bad a headache is awful? how drunk is incredible?
- This is really a third area of uncertainty – we have to standardize representations for these QUALITATIVE concepts

Qualitative Concepts

- Means there's no easy way to quantitatively order them, and often not even a agreed understanding of the meaning associated with the concept
- This agreement may not even be formally possible – e.g. what I call an awful headache might be trivial to somebody else, or vice versa….how do we know?
- This is an important issue, as communicating such concepts is very important (think of medical domains – how bad does this hurt?)

Fuzzy Concepts

- This is both a formal and informal term – when we talk about something being fuzzy it is a lesser-used synonym for vague
- Blurred at the edges really – think of Tall in this fashion
  - There are people that anybody would agree were tall – say somebody 8 ft tall.
  - There are also people that nobody would say was tall – say 2 ft tall.
  - The vast majority of us are somewhere in between
  - There are degrees of tallness, and we participate in the concept of tallness to some degree

Fuzzy Reasoning

- This is the basis for fuzzy reasoning – we participate in relationships to some degree
- More formally, fuzzy reasoning involves set theory, where an element has degrees of membership in the set instead of just being a member or not.
- We still have the usual problems associated with imprecision: if I say tall people are more likely to have heart attacks, there's imprecision in the rule itself, which has to be computed with the imprecision in the tallness of the subject…

Fuzzy Reasoning

- Some researchers do categorize fuzzy reasoning as outside of uncertain reasoning though
- Because it involved vagueness in the TERMS we use, not uncertainty about the world itself
- The practical problems are the same though – just a difference in the SOURCE of the uncertainty

Fuzzy Measures

- T(A) measures the degree to which A participates in relationship T, on a range from 0…1
- Combining these Measures: if T(A) measures the degree to which A participates in relationship T, then:
- T(A^B) = min(T(A),T(B);
- T(AvB) = max(T(A),T(B));
- T(~A) = 1-T(A)

Fuzzy Logic

- Is probably the most visible commercial application of uncertain reasoning
- Simply because there are so many areas of fuzziness in our perceptions of the real world
  - Video camera brightness levels
  - Automatic Transmissions in Automobiles
  - There are even "fuzzy logic vacuums" ???
  - Plenty of applicatons for this!
    - for example….

Fuzzy Logic in Robotic Control

- Fuzzy Logic is commonly used to represent imprecise concepts in robotic control
- Robotic control is a very interesting problem from an AI standpoint – want to do things at a high level – e.g. plan paths - yet have to deal with loads of low-level details
  - motors, sensors, and so on!
  - Latter emphasis requires a language that supplies lots of support for low level systems coding e.g. C/++

Fuzzy Logic in Robotic Control

- The problem is these aren't great languages for symbol manipulation/reasoning
- Similarly, lots of languages that specialize in symbol manipulation (e.g. Prolog) are really, really crappy at doing low-level systems work
- Most work in this area is done by adapting conventional languages to do more symbolic processing
- Or by taking very flexible symbol languages (e.g. lisp) and giving them systems level support

Client-Server Approaches

- Help to deal with this
- Robot acts as a server to a client controller program
- The robot would accept commands from your client software
- The robot would deal with the low level interface (e.g. *forward 100* becomes a motor on for a length of time, and recording the number of ticks passed in wheel encoders)
- We can also use a simulator to simulate the actions of a robot, allowing development of a client and testing on a software server to start with

Reactive Systems

- Involve simply reacting to external stimuli (we mentioned reactive agents before)
- These may have state to tie in previous reactions
- Still very limited
- Can react based on declarative rules (matching situations to perceptions, like we would in prolog)

Reactive Systems in Java

- Still, this is not easy to do in a conventional language like Java/C++ without extensive additions to the language
- No real facilities for manipulating symbols
- Symbol manipulation languages, on the other hand, tend to be poorer at these procedural elements
- The problem is there's still procedural stuff that has to happen: adjusting motors, reading sonars, and so on

Behaviour-Based Robotics

- Behaviours (also called Schemas) were developed as a mechanism to deal with the combination of low-level and high-level problems we see here
- A Behaviour is an activity package, with all the abilities necessary to accomplish some specific real world behaviour
- This is a nice way to look at things, because it echoes the physical world…
- For example, consider insects:

Insect Behaviours

- *Seek out potential food sources*
- *Seek out potential sexual partners (or attract them with displays of some sort)*
- *Attracted to light*
- *Flee from known predators*
- Each of these behaviours is ***always*** active, and has a strength associated with some motivation (e.g. hungrier -> more effort spent seeking out food) or perception (enemy closer – stronger urge to flee)
- The output of each behaviour is blended together (e.g. still attracted to light even though hungry) into an overall response for the insect's actuators (body)

For Example

- Consider moving around
- I want to move toward a goal (or keep moving if I don't have a specific goal)
- I also want to avoid bumping into things
- Or even avoid paths that will take me very close to obstacles
- I want to watch where I am going
- These can all be considered separate behaviours

Behaviour Combinations

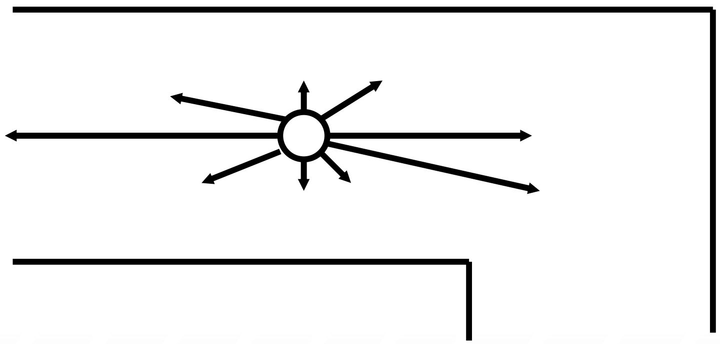
- The power of behaviours is in their combination
- Walking someplace, for example, involves ALL of those previous behaviours at the same time
- Different components are more or less important at different times but all play a role in influencing how we walk
- That is, our walking behaviour is composed of the collection of these lower level behaviours being performed in unison
- At some point, the lower level behaviours implement procedural changes – but the important stuff happens in the combination

Behaviour Desirability

- At any point, there may be a number of different things that could be done from the point of view of a collection behaviors
- One behavior may make us want to move forward, for example, while another may want us to keep our distance from obstacles
- We'll think of moving around, because the physical nature is easy to visualize. But the same thing can work with whatever else the agent is capable of
- We associate a desirability of each action in a particular situation with the potential output of a behavior

For Example…

- From the point of view of not avoiding walls, the length of the arrows below summarizes the desirability of proceeding in that direction:

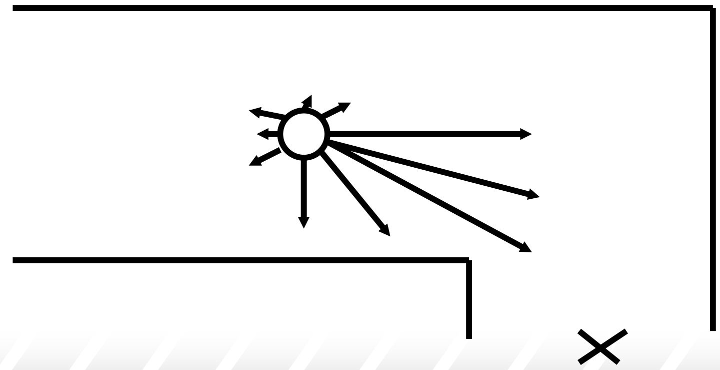


Measurements

- These could be a real measurement between 0 and 1, stored in a table or a function depending on which was convenient
  – both just mappings!
- Now visualize a desire to get to a certain location!

For Example…

- Now the green lines represent desirability of choices in terms of getting to the spot marked by X:

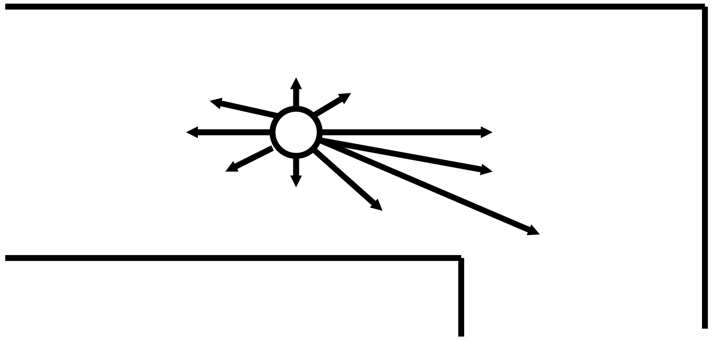


Each of These…

- Represents the potential responses for one specific behaviour- The interesting part is if we can put them together, and get goodness of getting to the X without bumping into walls
- The goodness of any choice would be a combination of the goodnesses from the point of view of each of the participant behaviours

Combination

- Combining the two would increase the goodness of those contributing to both, remove goodness from those detracting from both, and altering variously in between



Fuzzy Behaviours

- We can implement Fuzzy Behaviours that do just this
- Really there's two places fuzziness comes into play
- The invocation of a behaviour
  - e.g. attracted to light – how strong is the source? (fuzzy activation)
  - e.g. don't get too close – how close is too close?
- The strengths of the combined output
  - Taking the weights of each and giving a fuzzy response

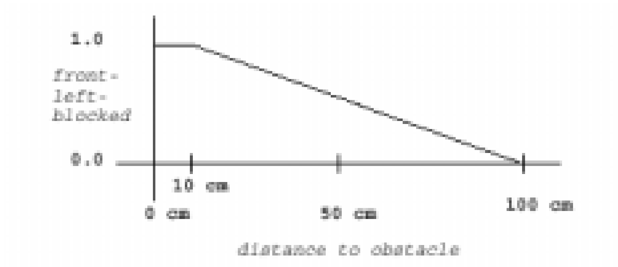
Each Behaviour

- Can contains a list of fuzzy variables (values between zero and one), each representing a fuzzy proposition of interest
- Every cycle, perception plugs values into these variables (get an overall fuzzy state)
- Each behaviour contains a rule set of the form: If Ai then Ci (for i=1..n)
- A is a logical formula based on the contents of fuzzy variables, C is a set of fuzzy control values to modify what the robot's doing

For example, Saphira

- A particular fuzzy architecture for robot control
- Kurt Konolige & Karen Myers, SRI. Off-the-shelf control methodology for Pioneer robots
- In Saphira, the fuzzy variables in behaviours are concepts like *front-left-blocked*, *side-right-blocked*, and so on
- A robot isn't usually *completely* blocked or unblocked – it's a question of degree, i.e. not having the desired amount of manoeuvring space
- Saphira provides built in membership functions to represent the fuzziness we want to associate with these variables

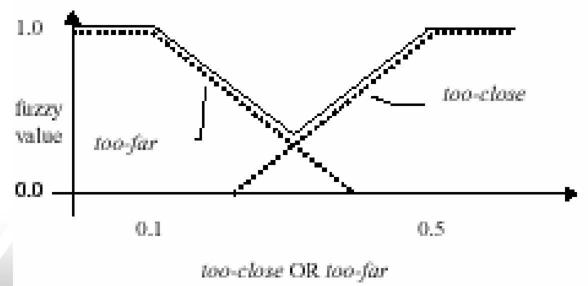
Front-Left-Blocked



- Front left becomes "less blocked" as we get closer to a 1m distance – more room for the robot to move
- Perception gets plugged into this fuzzy variable, and a response based on the shape of the function is generated

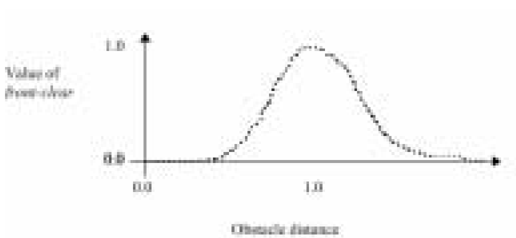
Membership Functions

- Saphira provides a library of these functions, and we can associate the shape we want with the fuzziness of variables we are interested in
- We can also create combinations with logical operators, which combine the functions according to the rules of fuzzy logic we saw earlier



Non-Linear Functions

- Membership functions don't have to be linear (they're just easier to piece together that way)
- e.g. membership in at-one-meter

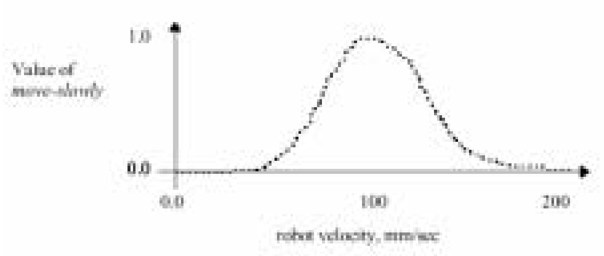


Rules

- We then create rule sets that define our behaviours, based on combinations of these fuzzy variables:
- IF front-left-blocked AND (not front-right-blocked) AND (NOT side-right-blocked) THEN turn-left
- Note that the response is also fuzzy – how much left? depends on ***how much*** we want to go left at the moment…and how much should be directly proportional to the strength of the conditions of the rule. That is -

Rule Conclusions

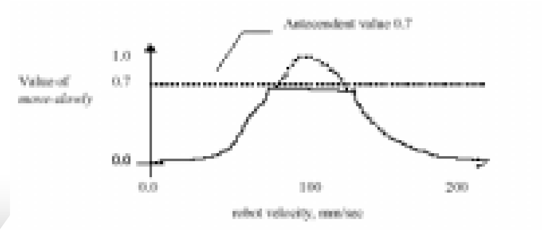
- are fuzzy control sets



- Move-slowly is a range of possible values centered around 100mm/s – the control value actually sent to the robot

Clipping based on Rule Antecedents

- This conclusion is adjusted based on the strength of the rule conditions (antecedents). A strength of 1 should not modify the control set; a strength of 0 should shut it off completely
- In between these, the antecedent of the rule forms a strength barrier on the strength of the rule's output (strength of the rule is the minimum of original o/p and antecedent's confidence)



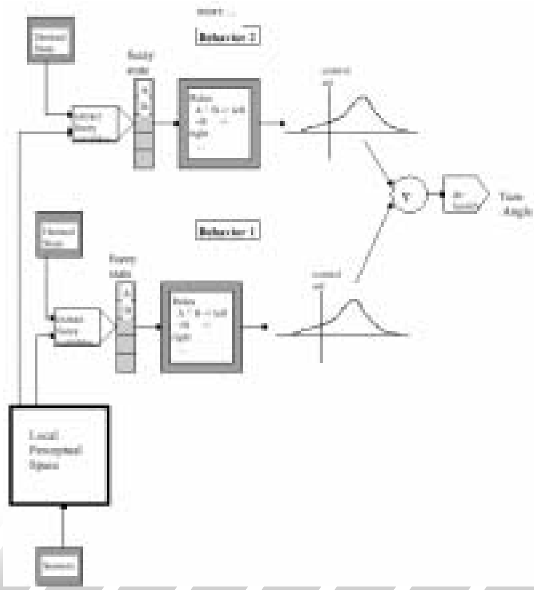
Processing Rules

- Each rule has a different set of fuzzy concepts – some may apply at the same time that reinforce or detract from one another
- In Saphira the control values of all the rules in a behaviour are averaged to produce a single desirability function as output

Behaviours Overall

- Consist of an update function and a set of these fuzzy rules
- The update function pulls the elements of perception a behaviour is interested in out of the total perceptions available from the robot
- A behaviour also has an importance that is set globally and establishes a priority for the behaviour
- The output of a behaviour is affected by the strengths of the rules (context-dependent) and this importance

The Big Picture

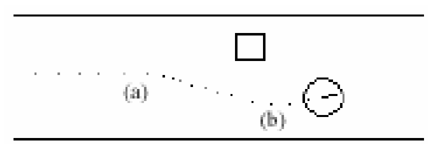


Integrating Behaviours

- Now, at each point in time each behaviour will have certain overall desired values for the possible control values on the robot that can be modified (e.g. speed, direction, gripper,…)
- We have to combine these responses, in a fuzzy manner, to get an overall change for the robot
- The amount contributed to the alteration of a robot's control values depends on the strength of activation of a behaviour
- Which in turn is controlled by it's overall importance and context-dependent activation of rules

Integrating Behaviors

- This allows (for example) goal-seeking behaviours (get here) to be combined rationally with reactive behaviours (stay away from walls!)



- For example, in a), goal directed behaviour dominates – as we get closer to the obstacle, the activation of the avoidance behaviour b) begins to dominate, then wanes as we pass the obstacle

Defuzzification

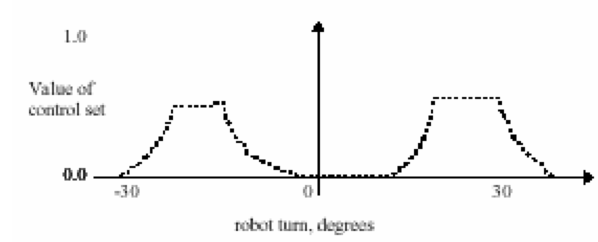
- We can blend rules within behaviours, and talk about the strength of combined behaviors for output
- In the end we still have fuzzy control sets
- We need to have a way of taking those and turning them into precise (crisp) values to send to the robot
- This process is called ***Defuzzification***
- There are lots of ways of defuzzifying
- We could just pick the control options with the highest activation (highest behaviour dominates)

Deffuzification

- There isn't one obvious peak though, and this is problematic, because slight changes in activation could lead to real swings in behaviour
- e.g. two similar peaks contributed by different behaviours, and we move from one to the other (thrashing between two behaviours)
- narrow high peaks also drop quickly, so we're surrounded by bad values – not particularly robust – imagine going through a narrow doorway where tiny variations could lead to a collision

Defuzzification

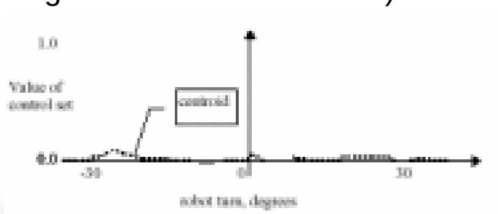
- We could also take the centroid of the set of control functions – the average weighted by the fuzzy values
- This is also problematic in some situations:



- here, average is going straight into a collision, rather than turning left or right

Saphira

- Uses a modified centroid approach (because where it isn't problematic it's robust and efficient)
- Where there isn't a single centroid, it looks at the total area underneath and picks the largest
- Also places a small peak at 0 to deal with the case where there is little info provided at all (in order that we don't continually adopt different behaviors when doing nothing makes the most sense)

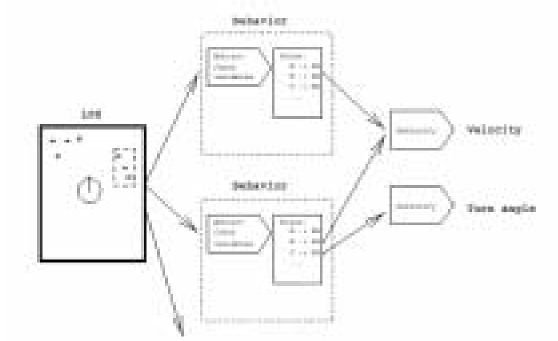


Actual Saphira Code

- For behaviours, is C like, but combines symbolic reasoning:

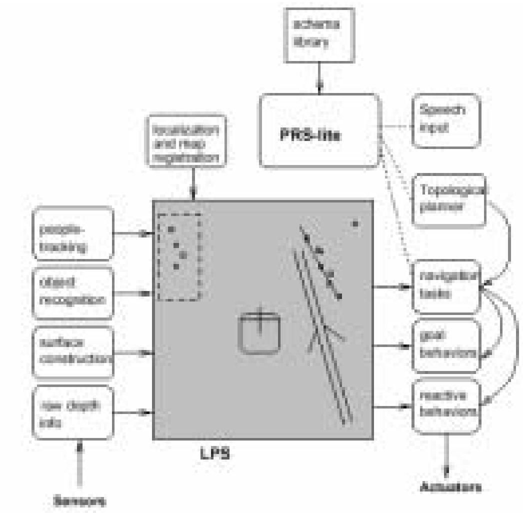
```
/*
 * Define the behavior constant_vel
 *   with one parameter, the desired speed in mm/sec
*/

BeginBehavior sfConstantVel    /* behavior name */
     Params sfFLOAT sval       /* input parameter, desired speed */
     Rules                     /* just one rule */
         If  too_fast Or too_slow  Then Speed sval
     Update                    /* calculate fuzzy variables here */
         too_fast = up_straight(current_vel(), sval, 50.0);
         too_slow = straight_down(current_vel(), sval, 50.0);
     Activity
         Speed 1.0
EndBehavior
```

Perception & World Modelling: LPS



- The LPS is also part of Saphira – it's a grid-based representation of the Local Perceptual Space – using sonar (in this case) to try to detect what parts of the world are solid and which can be moved through – forms the input to behaviours

Maintaining Perception



- lowest level of lps attempts to construct surfaces from sonar data (connect the dots)
- higher level object recognition attempts to match higher level structures (hallway, doors)
- Wheel odometry for localization

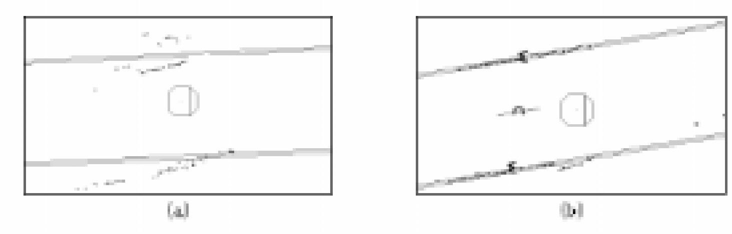
More Sophisticated Behaviours

- This is just a one-level approach
- Behaviours can be organized further though – envision this whole collection being a "wandering" behaviour
- This could be integrated with others at a high level as well
- Many schemes for this – subsumption architecture (Brooks), AuRa (Arkin),…
- Basically involves differences between how behaviours are related to one another, and decomposition methods (horizontal/vertical)

Also, Appreciate:

- This is just examining the basis in fuzzy control
- There are lots of side problems a system for mobile robotic control has to deal with…
- For example, we have to pull together perceptual information into objects (artifacts) we care about
- Saphira uses a probabilistic approach to look at how close sonar pings are together to try to find walls,gaps for doors, and so on
- When it finds them it records them in a data structure (an occupancy grid)
- We also now have the problem of -

Grounding



- This is actually a huge problem in perception
- Having seen something, how do we know how it changes? we *ground* or *anchor* our concepts of objects in an environment, and adjust those anchors as our perspective changes
- here, wheel encoders tell a robot how far it's moved – but they're inaccurate!

Grounding/Anchoring

- The robot has to continually adjust its world model (to the degree it keeps one) to reflect whether it's decided some new perception is a new view of an old object or not
- These errors compound hugely over time, and are part of why localization (Where Am I?) and navigation are such massive problems
- We can deal with this partly through multi-agent approaches (Blatant Self Promotion™)

Summary

- The same problems come up over and over in uncertainty
- no one scheme as the answers to all of them
- Some are more advantageous because of rigour and formality, others because they are more practically useful
- People in AI tend to be biased toward their favourites, but it's more important to be adaptable and recognize what might be useful somewhere!