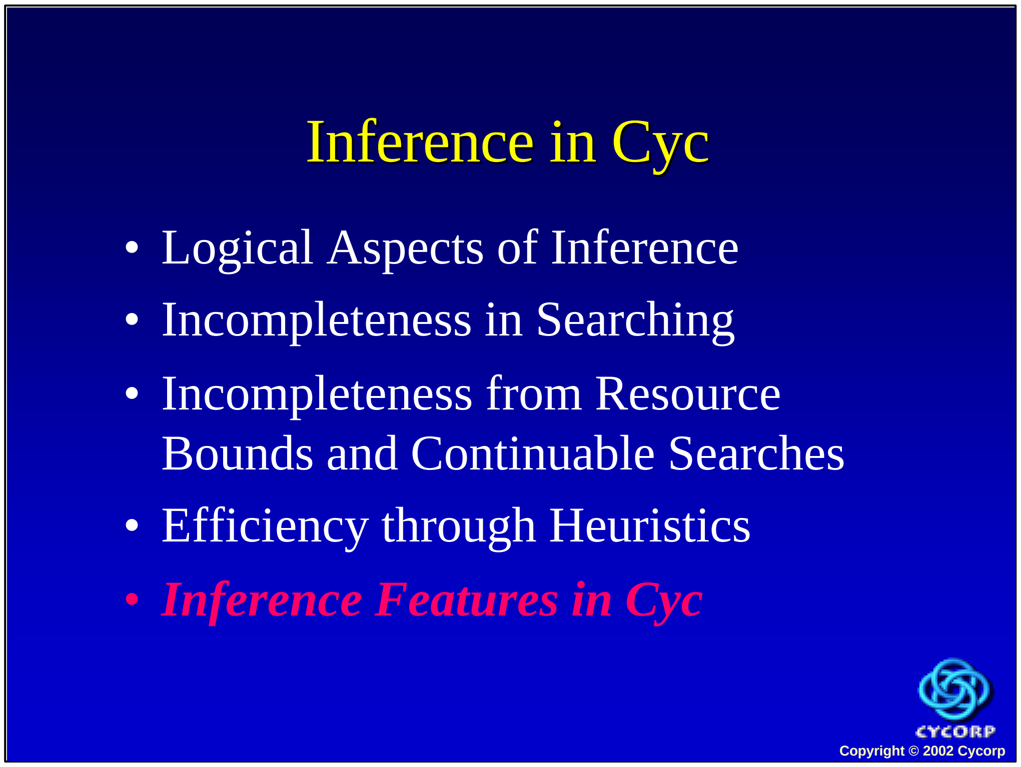
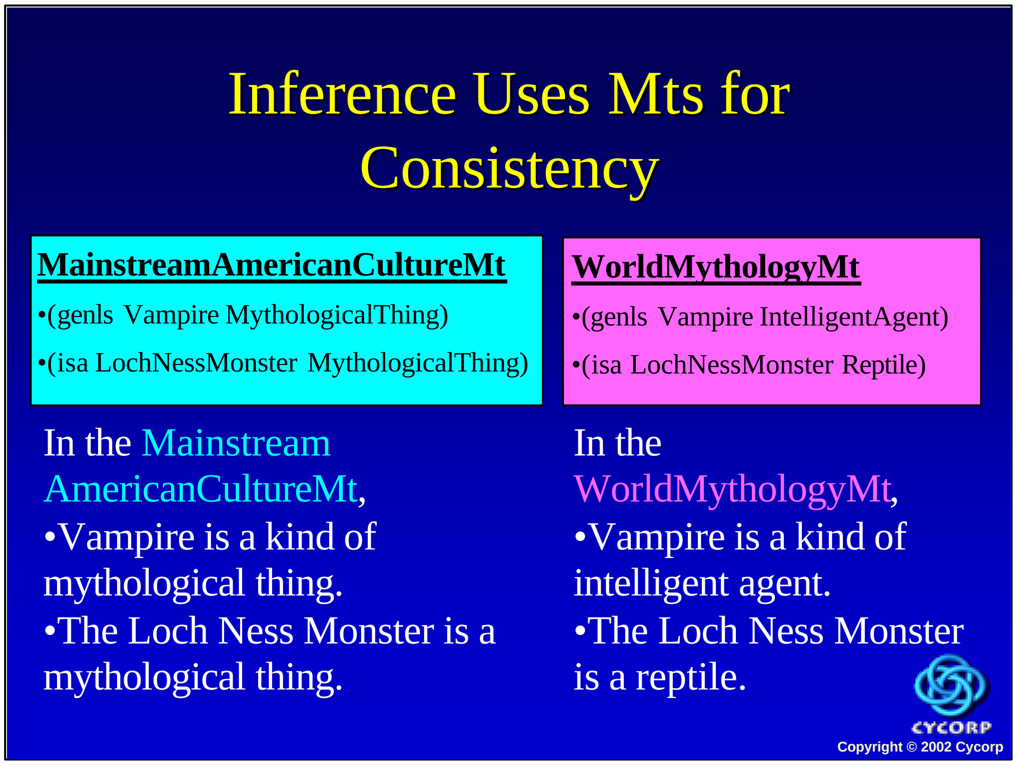


Inference in Cyc

- Logical Aspects of Inference
- Incompleteness in Searching
- Incompleteness from Resource Bounds and Continuable Searches
- Efficiency through Heuristics
- ***Inference Features in Cyc***

CYCORP

This is the final lesson in the Inference Tutorial. It will focus on microtheories and forward/backward inference.



Inference Uses Mts for Consistency

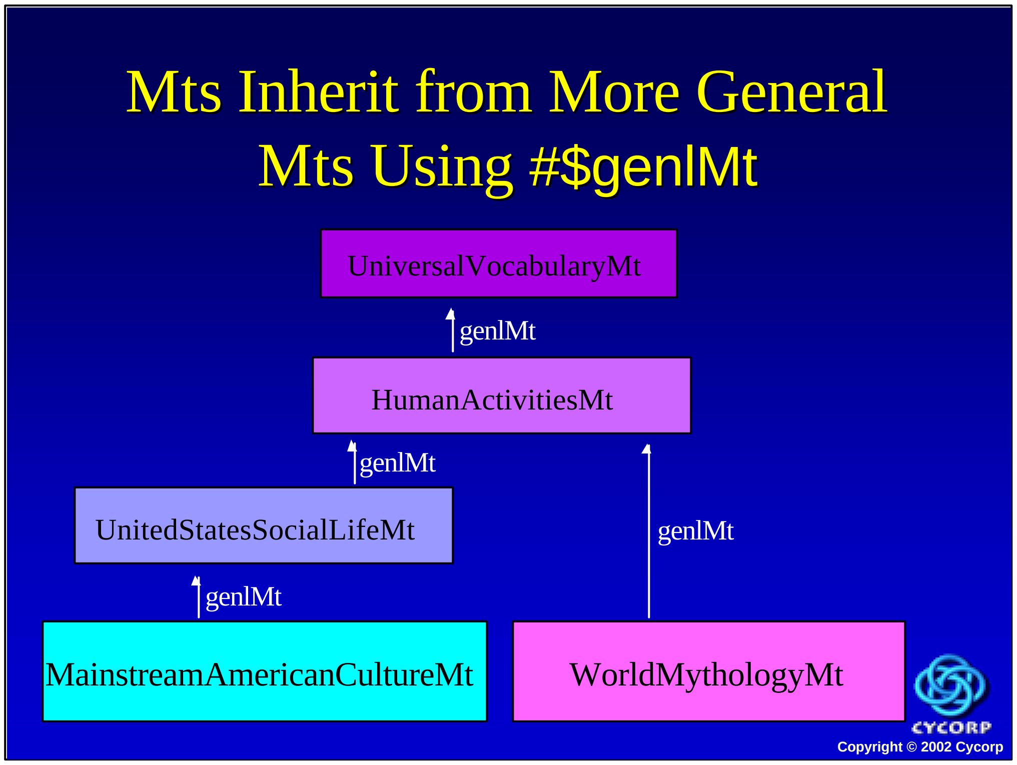
MainstreamAmericanCultureMt	**WorldMythologyMt**
•(genls Vampire MythologicalThing)	•(genls Vampire IntelligentAgent)
•(isa LochNessMonster MythologicalThing)	•(isa LochNessMonster Reptile)

In the MainstreamAmericanCultureMt,
- Vampire is a kind of mythological thing.
- The Loch Ness Monster is a mythological thing.

In the WorldMythologyMt,
- Vampire is a kind of intelligent agent.
- The Loch Ness Monster is a reptile.

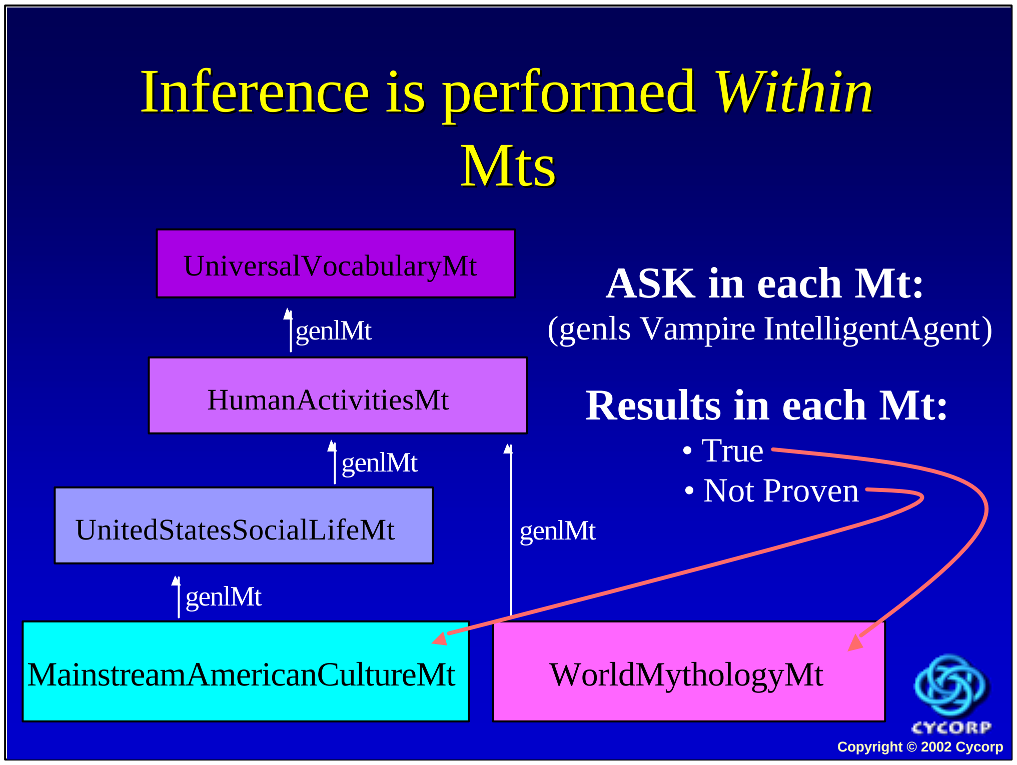
CYCORP

Another unique feature of the Cyc system is our use of microtheories to deal with the difficulty of having global consistency in a knowledge base. The Cyc Knowledge Base does not consist of one single theory that has to be consistent. As theories get larger and larger, it becomes more and more difficult to maintain consistency among all of the statements in them. We solved this problem in our system by not having just *one* theory; we have a large number of what we call "microtheories." These are smaller theories, usually on the order of a few hundred to a few thousand assertions in each one.



Mts Inherit from More General Mts Using #$genlMt

This gives us the ability to state that certain microtheories inherit from other microtheories; we can set up an ontology of *theories* and have one theory built upon other theories from which it inherits. It is easier to manage the space of millions of assertions because we carve them up into smaller sets of assertions that have common assumptions about them and then we state the relationships among these theories as a means of organizing them and making them more modular and reusable. The predicate that we use to state this inheritance relationship in microtheories is #$genlMt.

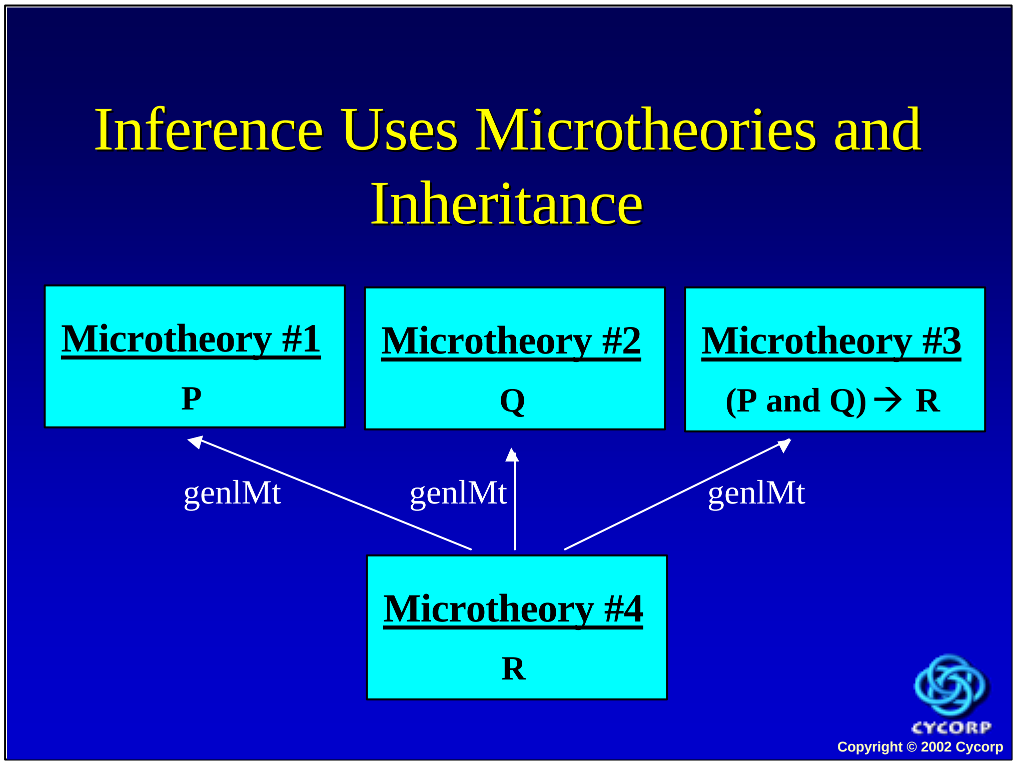


Inference is performed *Within* Mts

ASK in each Mt:
(genls Vampire IntelligentAgent)

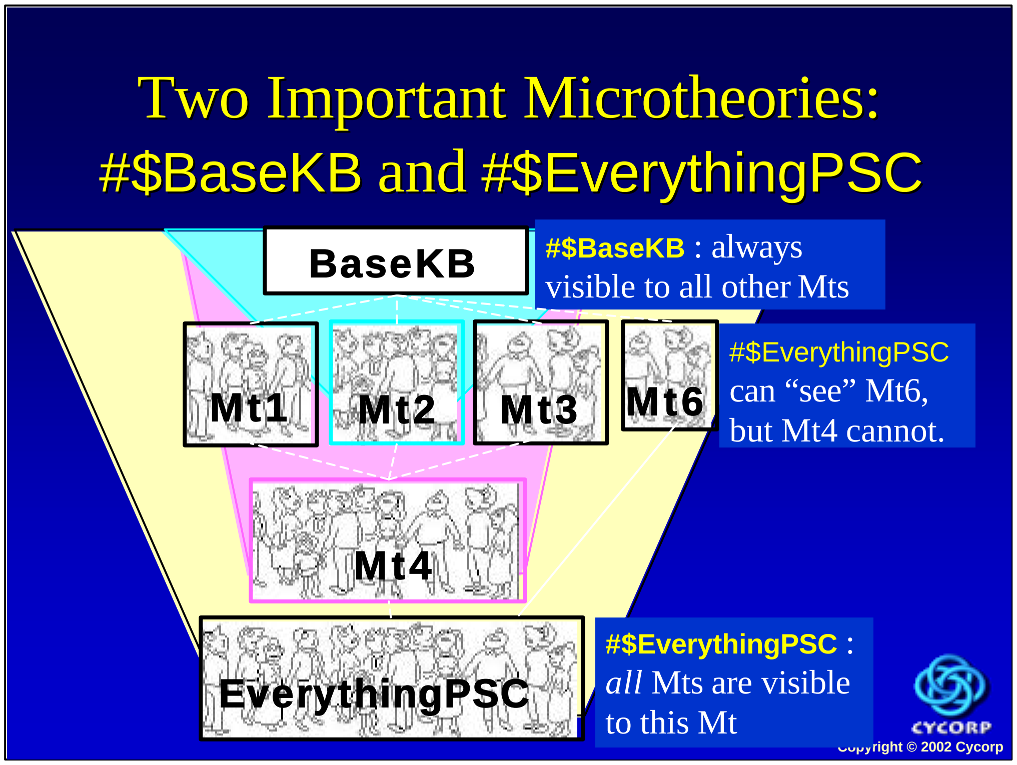
Results in each Mt:
- True
- Not Proven

When you perform an inference in our system, you perform an inference *in* a particular microtheory. This means that every assertion in that microtheory and all of the microtheories from which it inherits are “visible” for inference. This allows the system to maintain an enormous number of potential theories in an efficient fashion and to support performing inferences in any of them at the same time.



Inference Uses Microtheories and Inheritance

This slide shows a more complicated example of performing an inference within a microtheory. Imagine four microtheories, represented by the blue rectangles on the slide. In the three microtheories on top, there are three assertions. In the first microtheory, there is the assertion we're calling *P*. In the second microtheory, there is the assertion called *Q*. In the third microtheory, we have a rule that says that *P* and *Q* together imply *R*. Notice that in none of these three microtheories do we have a theory from which you can conclude *R*. But the fourth microtheory, below, since it has the three microtheories above it as genMt's, in effect inherits all of those assertions in one place. So in that microtheory you now have a theory which can see all of those three assertions and therefore soundly logically can deduce *R*. If you were to ask *R* in any of the three theories above, Cyc *would not* be able to prove it. But if you were to ask it in the theory below, you *would* be able to prove it.



Two Important Microtheories: #$BaseKB and #$EverythingPSC

#$BaseKB : always visible to all other Mts

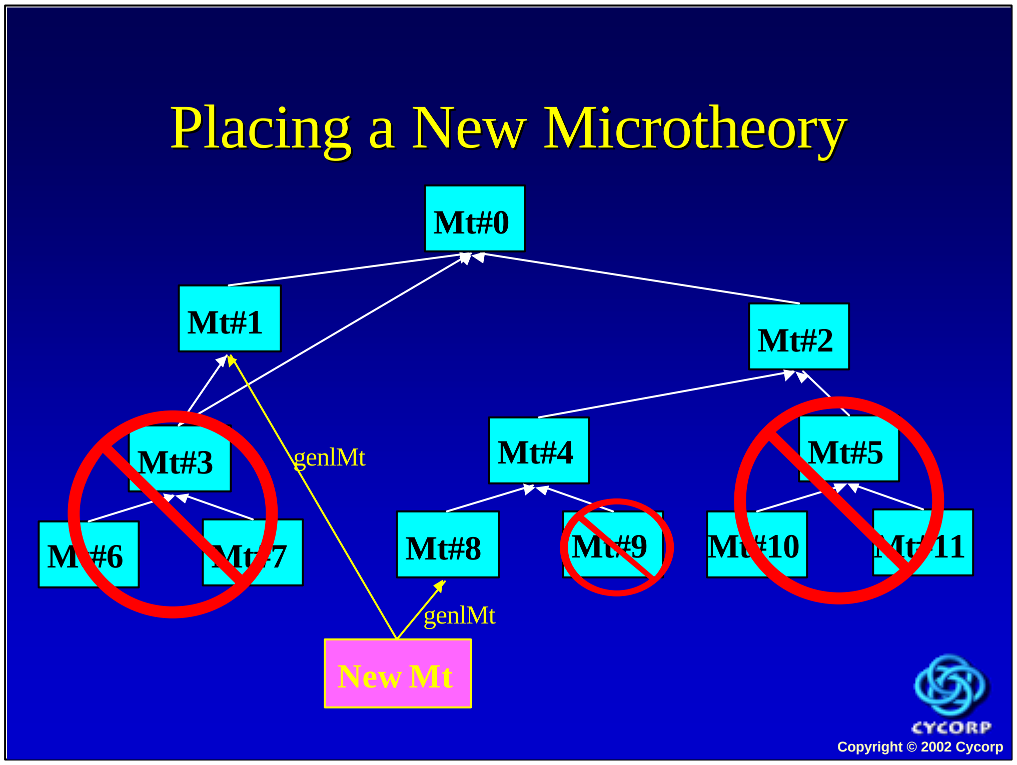
#$EverythingPSC can "see" Mt6, but Mt4 cannot.

#$EverythingPSC : *all* Mts are visible to this Mt

There are two microtheories which are worth pointing out as being interesting in cases like the one depicted on the previous slide.

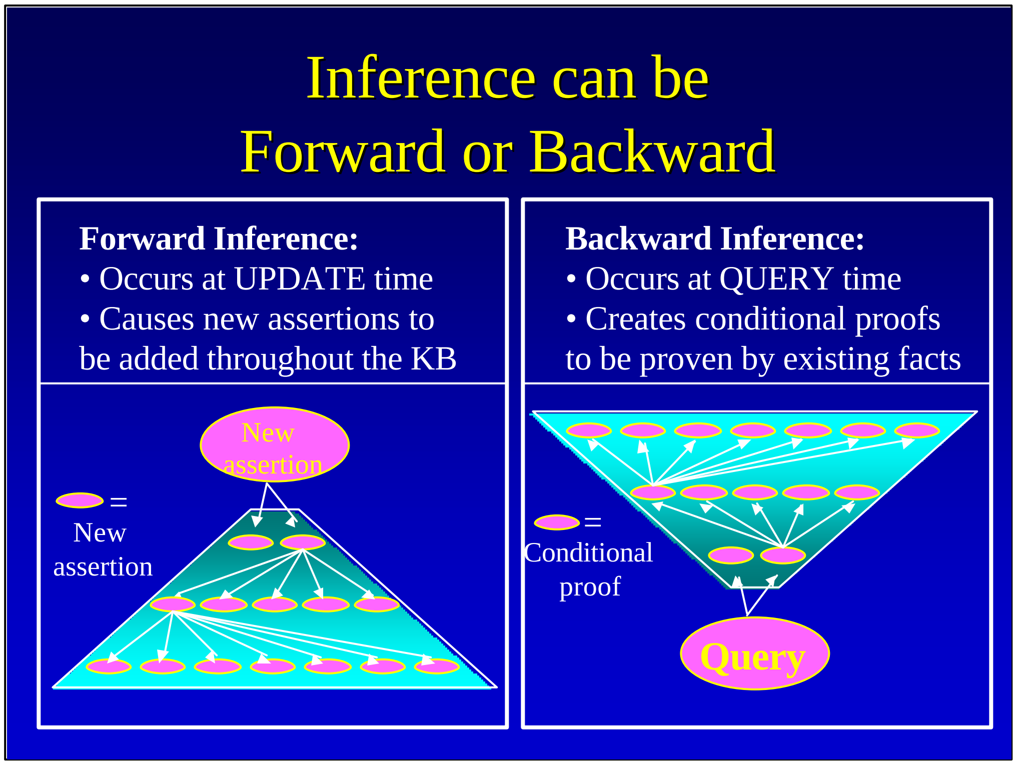
The #$BaseKB can thought of as the microtheory on top, from which everything inherits. So this microtheory is always visible to all other theories. It's meant to represent the universal theory vocabulary -- everything which is true, no matter what the theory. In fact, there are now approximately six microtheories "above" #$BaseKB (an example is #$UniversalVocabularyMt). But it is still true that all microtheories (except for these six) can see #$BaseKB (and, as a result, can see the all of the microtheories above #$BaseKB.

The converse of #$BaseKB is a microtheory called #$EverythingPSC (PSC stands for "Problem Solving Context".). This can be thought of as the bottom of the microtheory ontology, which inherits from every microtheory in the system. In general, this is not a sound thing to do. But for pragmatic reasons, in various applications it is often useful to have available a microtheory in which you can do an ask that will effectively ignore all of the other microtheories. #$EverythingPSC is a microtheory which has no logically consistent meaning but has a high practical utility just because it is able to see the assertions in *every* microtheory.



Placing a New Microtheory

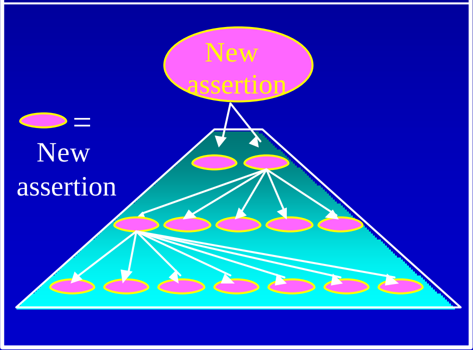
So, when you're designing an application, it is useful to introduce a microtheory into the ontology of microtheories and add some judicious #$genlMt links to the theories that you want to use. You are, in effect, constructing the theory you want your application to reason in. This allows you to control which theories you use in inference and which theories you ignore, providing another mechanism for filtering out and pruning the space of possible proofs that you'd make when you're performing inferences.

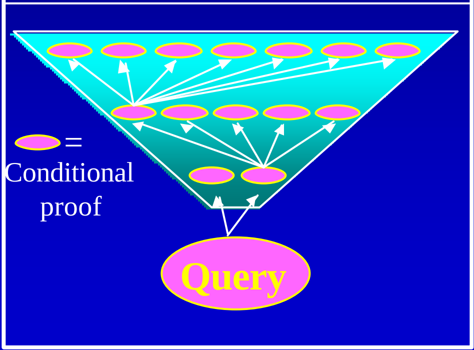


Inference can be Forward or Backward

Forward Inference:
• Occurs at UPDATE time
• Causes new assertions to be added throughout the KB

Backward Inference:
• Occurs at QUERY time
• Creates conditional proofs to be proven by existing facts

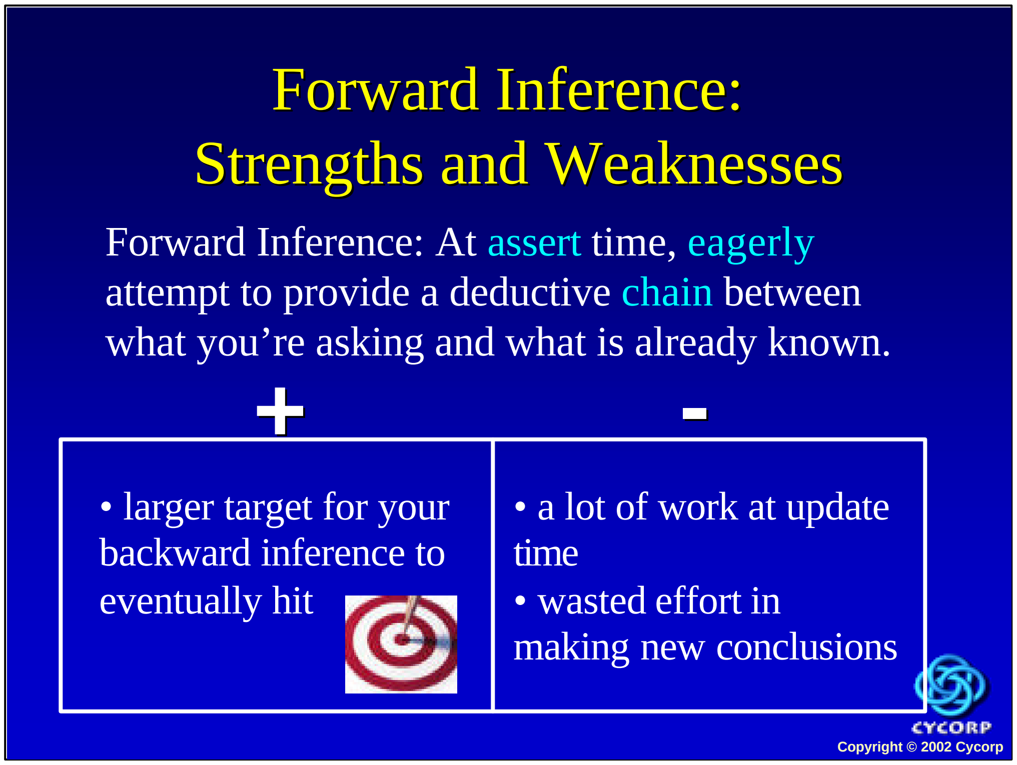




Inference in Cyc is not limited to either forward or backward inference. We support both. Let me describe what we mean by “forward inference” and “backward inference.”

Forward inference can be considered eagerly concluding additional assertions as soon as new assertions are added to the system. So forward inference occurs at update time to the Knowledge Base. In effect, it causes more updates to the Knowledge Base which then cause more updates, until eventually it ends the system by allowing operations to complete normally. So, forward inference is eagerly concluding from the assertions towards new assertions that you may or may not ever want to use in inference.

The opposite of forward inference is backward inference. Backward inference occurs at query time and starts from particular queries that you want to ask. It attempts to prove mechanisms for how the original query would be true in terms of something else and hopefully you can chain these conditional proofs back until you eventually hit something which already is true in your knowledge base and stop the backward search.



Forward Inference: Strengths and Weaknesses

Forward Inference: At assert time, eagerly attempt to provide a deductive chain between what you're asking and what is already known.

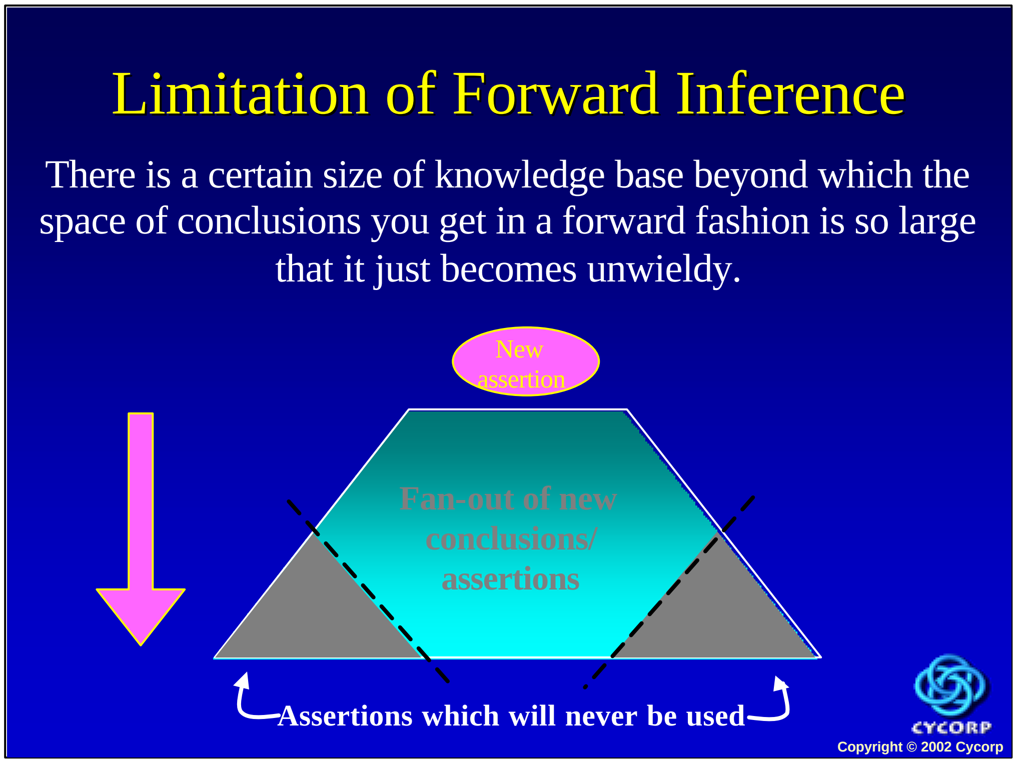
+	-
• larger target for your backward inference to eventually hit	• a lot of work at update time • wasted effort in making new conclusions

CYCORP

Copyright © 2002 Cycorp

Both forward and backward inference can be thought of as an attempt to provide a deductive chain between what you're trying to ask in your query and what you have already known in your assertions. So, you want to find a connection between these two? They can be thought of as just two different approaches to doing inference. One is do it eagerly at assert time and one is do it lazily at query time.

There are strengths and weaknesses to both. The strength of forward inference is that it provides a larger target for your backward inference to eventually hit. But the weakness is that you have to do a lot of work at update time. So if you have a lot of forward inference, the amount of work you do at update time could be quite large -- it could get larger and larger and larger as the knowledge base grows, so that it could eventually reach a point where there is so much to do at update time that you can't keep up with the updates. There's a limit to how much you can do with strictly forward inference in Cyc, because the space of potential things you can conclude is truly large and it's often far larger than the space of things that you ever want to actually ask the system.



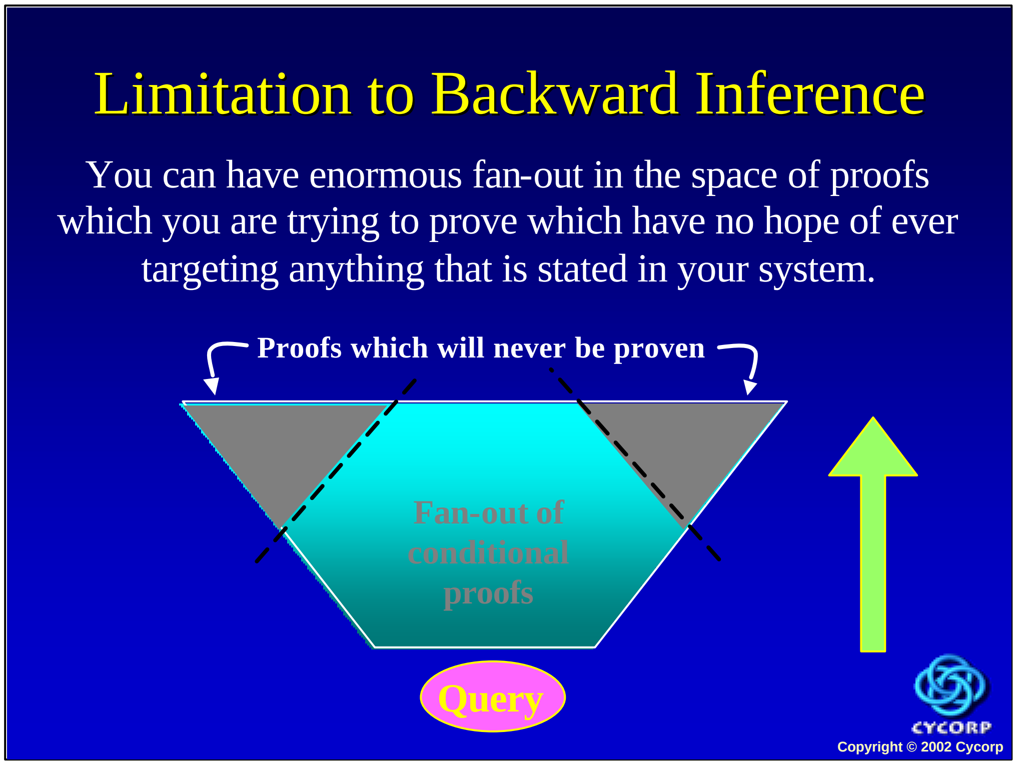
Limitation of Forward Inference

There is a certain size of knowledge base beyond which the space of conclusions you get in a forward fashion is so large that it just becomes unwieldy.

New assertion
Fan-out of new conclusions/ assertions
Assertions which will never be used

In a system with exclusively forward inference, you would have a lot of wasted space spent on concluding things that you aren't ever going to ask about, which in the diagram on the slide is like the two triangles at the bottom which indicate things you've bothered to conclude but are never going to bother to ask about.

Systems that have exclusively forward inference are fairly common in other knowledge representation systems. You can think of active databases with triggers as being ones that are exclusively forward. There are other well-known representation systems in the AI community: the RETE match is an exclusively forward-matching strategy. Magic Sets Transformation in the AI literature talks about how to encode backward inference in an exclusively forward system. So, there are many systems out there that are exclusively forward, and the limitation of them is that there is a certain size of knowledge base beyond which the space of conclusions you get in a forward fashion is so large that it just becomes unwieldy.



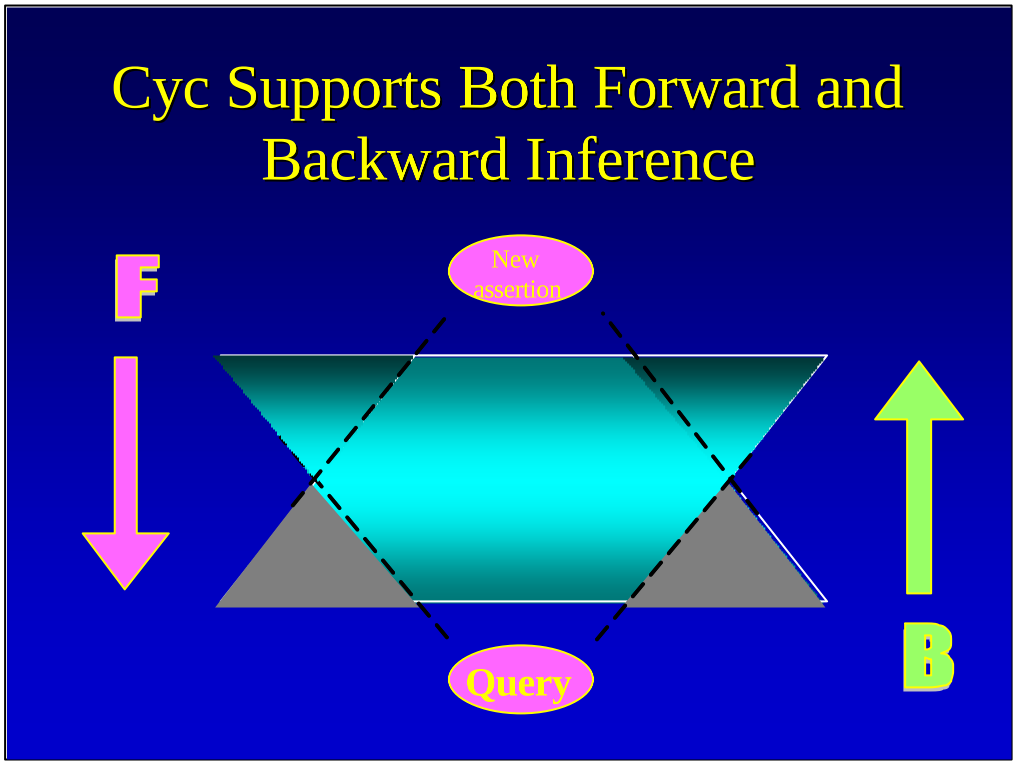
Limitation to Backward Inference

You can have enormous fan-out in the space of proofs which you are trying to prove which have no hope of ever targeting anything that is stated in your system.

Proofs which will never be proven
Fan-out of conditional proofs
Query

Backward inference is another common strategy which is often exclusively used in other systems. In backward inference you don't try to remember anything beyond what is stated to the system and you re-derive things when asked at query time. Exclusively backward systems are those like Prolog, where the set of rules and facts are stated to the system ahead of time and proofs are done exclusively at query time, in a backwards fashion; and if you want to re-prove it, you have to re-run the proof again.

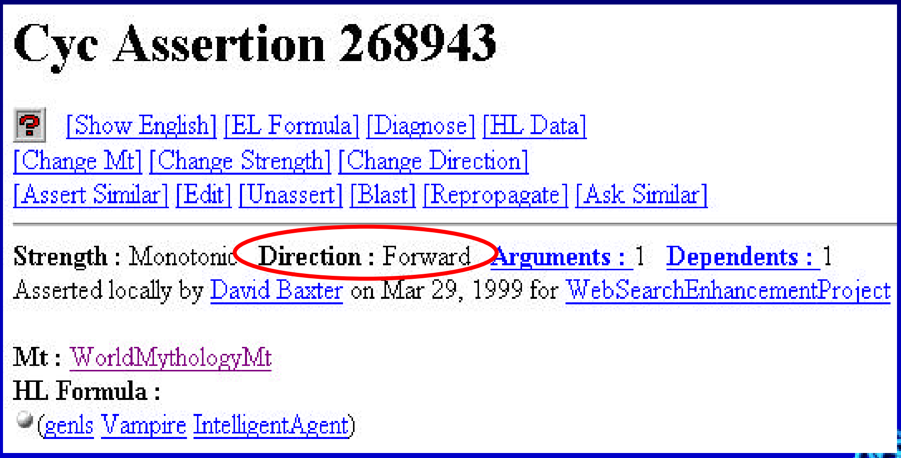
The downside of an exclusively backward system is the flip-side of that of the exclusively forward system. You can have enormous fan-out in the space of proofs which you are trying to prove which are never going to bottom-out at anything you know about. In this diagram, that is equivalent to the triangles on the top, which represent queries fanning out from the query that you asked, that have no hope of ever targeting anything that is stated in your system.



Cyc Supports Both Forward and Backward Inference

The benefit of having a system which supports both forward and backward inference is that with a judicious amount of forward inference you can increase the target of knowledge that is already represented in the system so that you have a larger target for your backward inference to hit. In the diagram on this slide, that is represented by the two approaches judiciously meeting in the middle. So, you can save all of the wasted space in the triangles by using a judicious amount of forward inference to expand the target area for your backward inference to hit.

A Subset of the KB is Marked "Forward"



Cyc Assertion 268943

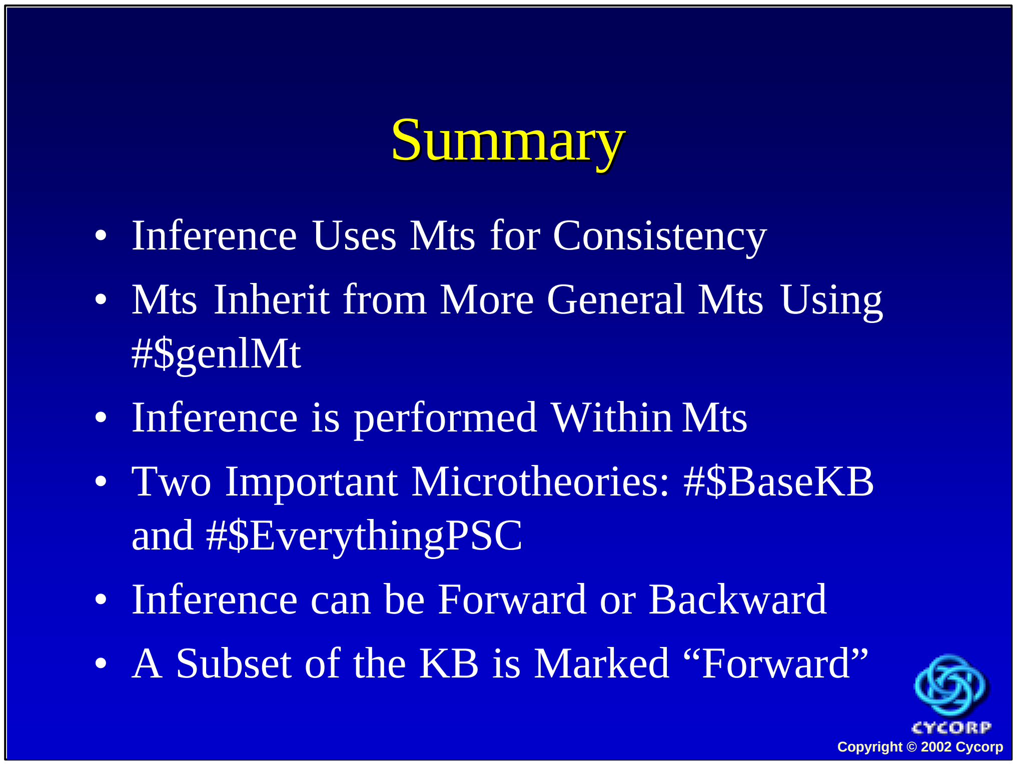
[Show English] [EL Formula] [Diagnose] [HL Data]
[Change Mt] [Change Strength] [Change Direction]
[Assert Similar] [Edit] [Unassert] [Blast] [Repropagate] [Ask Similar]

Strength : Monotonic **Direction :** Forward **Arguments :** 1 **Dependents :** 1
Asserted locally by David Baxter on Mar 29, 1999 for WebSearchEnhancementProject

Mt : WorldMythologyMt
HL Formula :
(genls Vampire IntelligentAgent)

Cyc supports both forward and backward inference, and this is the way it is used: every assertion in the system is labeled as being either a forward assertion or not. You can think of all of the forward assertions in the knowledge base as being a subset of the knowledge base that is labeled "forward", and amongst all of the forward assertions, whenever a new assertion comes in, forward inference triggers and runs exhaustively amongst just that set. By judiciously choosing a subset of the knowledge base on which it is worthwhile to perform this forward inference, we can have a good mixture of the benefits of both forward and backward inference without having to suffer through the weaknesses of having only one or the other.

Just to give you an idea of what is labeled "forward" in the system, effectively all GAFs in the system are labeled "forward" and a tiny percentage (probably around 5 percent or less) of the rules in the system are labeled "forward". The kind of rules that are labeled forward are typically those which are either extremely application-specific or constraints of some kind. The application-specific rules are in some focused microtheory so that the application knows that it wants these conclusions done because it is going to target exactly the results of those conclusions. Things like arities and #$argType constraints (and some classification rules that conclude things are instances of other things) are worth doing in a forward fashion -- especially those things that have to do with canonicalization and well-formedness checking; these are things where you don't want to do deep inference at assert time to check those things, so it's good to have those things computed in a forward fashion so that you can have simpler queries in the system to check them.



Summary

- Inference Uses Mts for Consistency
- Mts Inherit from More General Mts Using #$genlMt
- Inference is performed Within Mts
- Two Important Microtheories: #$BaseKB and #$EverythingPSC
- Inference can be Forward or Backward
- A Subset of the KB is Marked “Forward”

CYCORP

This concludes the tutorial on Inference in Cyc.