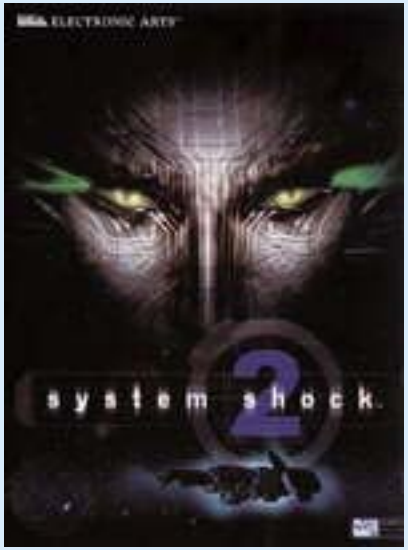
Meeting Players Half Way

Using Adaptive Methods to Prevent Player Frustration

Irrational Games



Accessible

Not Dumbed Down

Adaptive Training

- Don’t train things the players knows
- Teach players when they screw up
- Help you pick up where you left off

Problem: Games are too complex

Solution: Training Sequences

Now you've got two problems.

Things To Train

- Gameplay Conventions
- Controller Conventions
- Gameplay unique to each game
- Strategy unique to each game

Gameplay Conventions

FPS Controller Conventions

- Jump on face button
- Crouch by clicking movement stick
- Right trigger shoots

Conventions

- Instantly familiar
- Learn once, apply for many games
- Do you train conventions?

Training Sequences

- Too Few
  - Player doesn’t know conventions
  - Player feels lost
  - Player miss depth of the game
- Too Many
  - Click through
  - Annoyed and fraustraing first experience

Ideally…

- Beginning of the game
- Should be exciting
- Only Introduce the major unique gameplay

Adaptive Training Goals

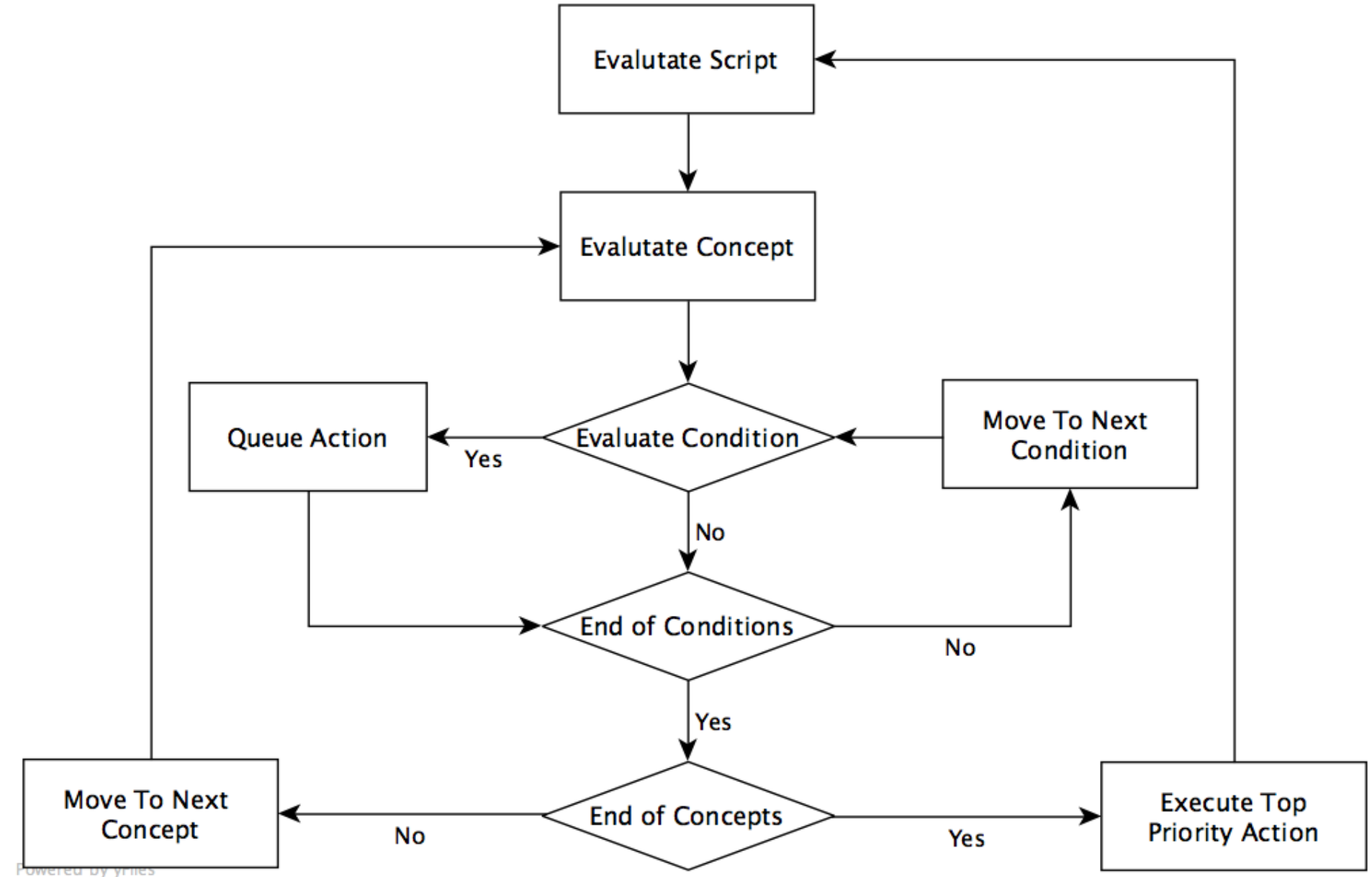
- Complements linear training sequence
- No more, “Here is how to jump, Marine”
- Wider range of messages
  - Strategy
  - Hints
- Tool tips for gameplay

Expert Systems

- Designer brain in a box
- Capture expert knowledge in a narrow domain
- Wide Range of Applications
  - Medical Diagnosis
  - Accounting (Tax Advisors)
  - Tutoring

Bioshock Training Script

- List of Concepts
  - List of Conditions - IF-THEN Rules
  - Triggers Training Messages
- Conditions only test things in a Fact Database
- Forward Chaining Inferencing


Evalutate Script
Evalutate Concept
Queue Action
Yes
Evaluate Condition
Move To Next Condition
No
End of Conditions
No
Yes
Move To Next Concept
No
End of Concepts
Yes
Execute Top Priority Action

Infinity Engine Scripting

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

```
IF
    Class(LastAttackerOf(Myself), MAGE)
    HPGT(Myself,50)
THEN
    RESPONSE #80
        Attack(LastAttackerOf(Myself),MELEE)
    RESPONSE #40
        Help()
        RunAway()
END
IF
    Exists(LastAttackerOf(ProtectedBy(Myself))
THEN
    RESPONSE #100
        Attack(LastAttackerOf(ProtectedBy(Myself))
                , RANGED)
END
```

Final Fantasy XII - Gambits

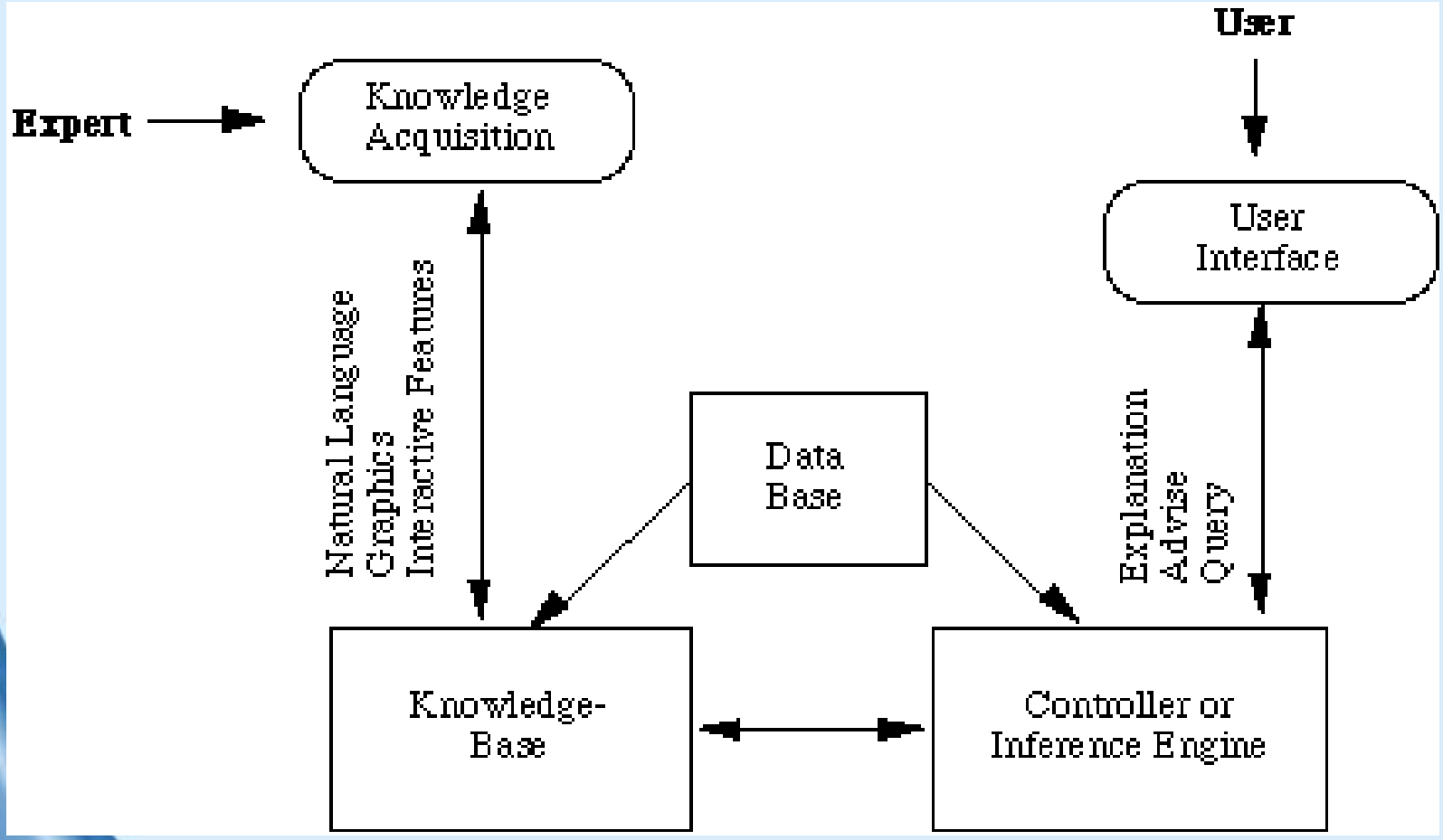
QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

Gambit into psudo-code

```
IF
   dead(allies) and has(pheonix_down)
THEN
   Use(pheonix_down, dead(allies))

IF
   poisoned(allies) and can_cast(poisona)
THEN
   cast(poisona, poisoned(allies))
```

Structure of an Expert System



Knowledge Acquisition

- Gambits are created by an interface by the User
- Bioshock Training Script created by designer through a visual scripting system.

Knowledge Base

- If - Then rules
- Gathered from the experts, either directly or through a knowledge engineer
- Models processes and heuristics of experts

Inference Engine

- Backwards Chaining
  - Given a goal and reach it by deriving facts
- Forward Chaining
  - Reach conclusions given facts

Backwards Chaining

- Goal Driven
- Structured Selection
  - Find best diagnosis
  - Identification
- Can gather data as needed

How it works

- Given list of goals
- Assume Then part
- Try to prove If part
- Try to prove family is albatross

IF
family is albatross and
color is white

THEN
bird is laysan albatross.

IF
family is albatross and
color is dark

THEN
bird is black footed albatross.

Forward Chaining

- Data driven
- Infer new facts based on current data
- Keep track of current state of inference
- Uses a Fact Database

Example

- Data
  - Alice is married to Bob
  - Bob is Ken’s father
- IF-THEN Rule
  - If X is married to Y and Y is Z's father then X is Z's mother
- Now this fact can be used in another rule

Fact Database

- Conditions use fact to determine game state
- Game update facts when needed
- Benefits of separation
  - Uniformity of rules
  - Optimization (Rete)
  - Ease of testing

What’s a Fact?

- (x, on, y)
- Three Slots
- Represents relations, or objects
- String facts for more slots
- (vector, 1, 2, 3) = (vector, 1, vector123), (vector123, 2, 3)

Pattern Matching

- Wild cards
  - ?x named variable
  - IF (?x, on, ?x) THEN assert(?x can’t be on itself)
  - ? unnamed wild card, if you don’t need the value

Using wild cards

- Inference
  - (Alice, Married, Bob)
  - (Bob, Father, Ken)
  - if (?x, Married, ?y) and (?y, Father, ?z) then assert (?x, Mother, ?z)
- String facts
  - if (vector, ?x, ?link) and (?link, ?y, ?z)

Gameplay Example

- Security Systems
  - Cameras can see you
  - After they spot you they will trigger alarm
  - You can evade them or shoot them to stop triggering of alarm
  - You can stop alarm by finding a security station

Concepts

- Represents a particular aspect of gameplay
  - How to use weapons effectively
  - What to do next in a quest
  - You can turn off alarms
- Knowledge level
  - Models if player understands the concept
    - -1 is player doesn't understand the concept
    - 1 is player understands the concept

Conditions

- A If-Then rule that can affect the understanding of concepts
- Example:
  - If player has triggered alarm then change knowledge of Security Alarm by -.1
  - If player has shutdown security then change knowledge of Security Alarm by .5

Fact Design

- Balance of designer and programmer
  - logic in condition vs when to assert facts
- Need clear communication of assert vs retract
- Avoid testing if fact is not true
  - AlarmOn
  - AlarmOff

Message Triggers

- Display a training message
- Triggered by knowledge level changes
- Can have different levels of training
- Example
  - When knowledge level is -.3, show message telling you to avoid cameras
  - When knowledge level is -.6, show modal tutorial screen with details about the system.

Knowledge Updates

- Bayesian
  - Used in tutoring systems, each problem can be wrong due to multiple failure conditions
- Linear
  - Easier to understand and reason with
  - Few updating rules,
  - unambiguous failures

Implementation

- Modified Unreal 3 Engine
- Uses a visual scripting system based in UnrealED
- Designer already knows the system
- Design Pattern : Interpreter
- Only need to provided Facts in game code

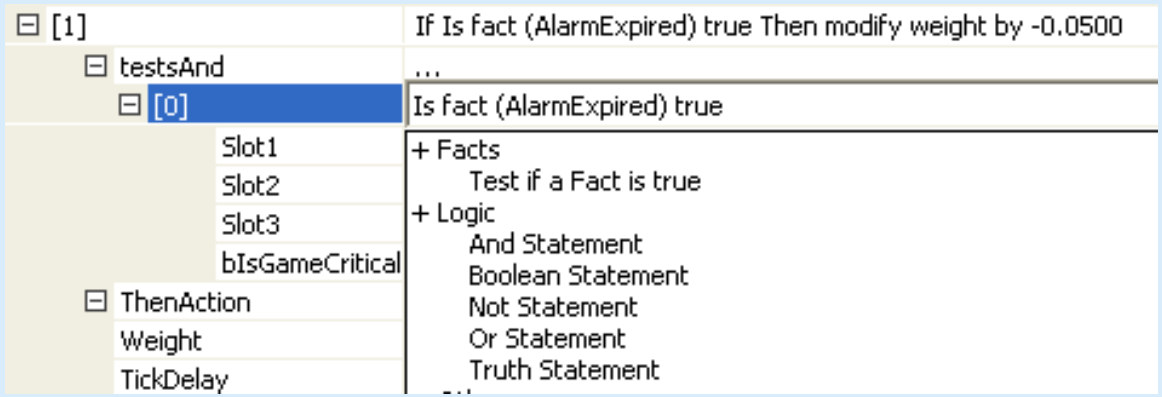
Sample Script

⊟ Concepts	…
⊟ [0]	Concept SecurityCameraOverall with initial knowledge of 0.0000
ConceptName	**SecurityCameraOverall**
KnowledgeLevel	0.000000
⊞ Conditions	…
⊞ MessageTriggers	…
enabled	True
bIsGameCritical	False
⊟ [1]	Concept SecurityCamerasAvoid with initial knowledge of 0.0000
ConceptName	**SecurityCamerasAvoid**
KnowledgeLevel	0.000000
⊞ Conditions	…
⊞ MessageTriggers	…
enabled	True
bIsGameCritical	False
⊟ [2]	Concept SecurityStations with initial knowledge of 0.0000
ConceptName	**SecurityStations**
KnowledgeLevel	0.000000
⊟ Conditions	…
⊞ [0]	If Is fact (AlarmCancelled) true Then modify weight by 0.2000
⊞ [1]	If Is fact (AlarmExpired) true Then modify weight by -0.0500
⊞ MessageTriggers	…
enabled	True
bIsGameCritical	False
⊞ [3]	Concept SecurityTimers with initial knowledge of 0.0000

- Training Script
  - Array of Concepts
  - Agenda: prioritized list of activated conditions
- Concept
  - Knowledge level
  - An array of conditions
  - An array of message triggers

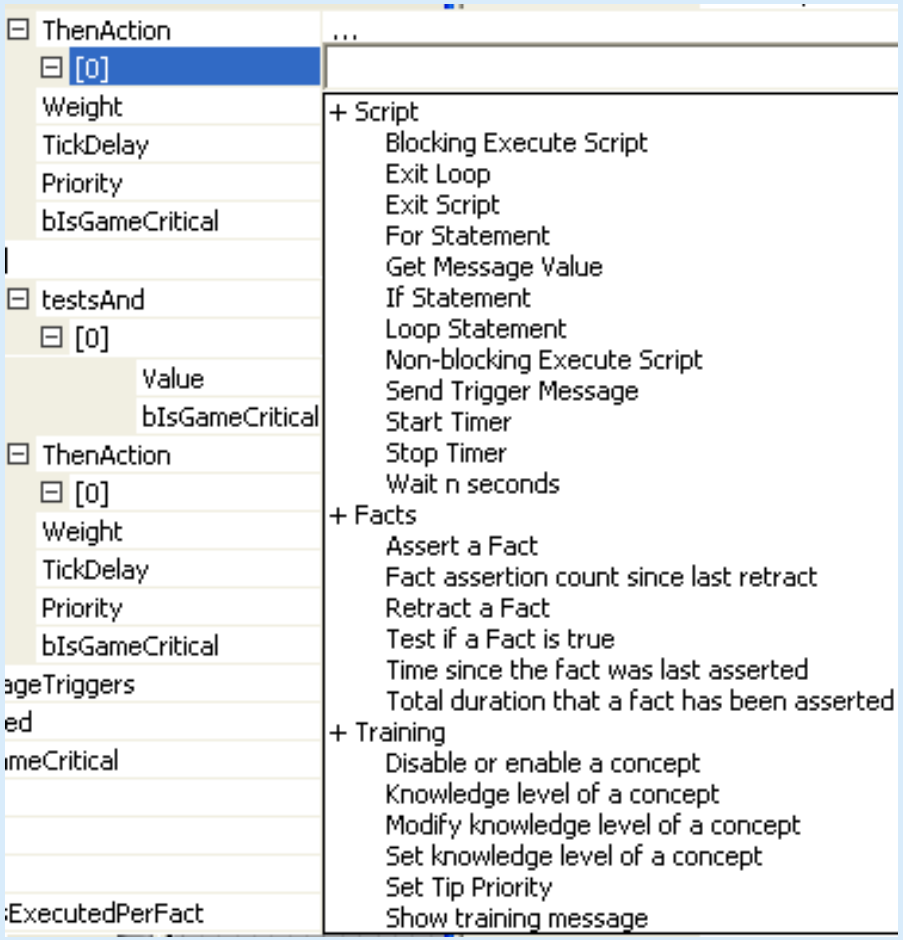
⊟ Conditions	…
⊟ [0]	If Is fact (AlarmCancelled) true Then modify weight by 0.2000
⊟ testsAnd	…
⊞ [0]	Is fact (AlarmCancelled) true
⊞ ThenAction	…
Weight	**0.200000**
TickDelay	**17**
Priority	0
bIsGameCritical	False
⊟ [1]	If Is fact (AlarmExpired) true Then modify weight by -0.0500
⊟ testsAnd	…
⊞ [0]	Is fact (AlarmExpired) true
⊟ ThenAction	…
Weight	**-0.050000**
TickDelay	**30**
Priority	0
bIsGameCritical	False

- Condition
  - Array of Action with results anded
  - Array of Array of actions to perform if true
  - Weight: How much to modify knowledge
  - Priority: position in Agenda
  - TickDelay: A hack to improve performance



- Filter actions based on return type
- Logic expression actions

General Actions



Fact Actions

- Operations
  - Assert : Allows for forward chaining
  - Retract
- Properties
  - Number of times of assert since last retract
  - Time since last retract
  - Time since last assert

More Complex Example

⊟ [1]	Concept CorrectAmmoType with initial knowledge of 0.0000
ConceptName	**CorrectAmmoType**
KnowledgeLevel	0.000000
⊟ Conditions	...
⊞ [0]	If Is fact (GoodAmmoUsed) true Then modify weight by 0.2000
⊞ [1]	If Is fact (OKAmmoUsed) true AND NOT(Is fact (GoodAmmoAvailable) true) Then modify weight by 0.0500
⊟ [2]	If Is fact (BadAmmoUsed) true AND Is fact (OKAmmoAvailable) true AND NOT(Is fact (GoodAmmoAvailable) true) Then do 4 actions
⊟ testsAnd	...
⊞ [0]	Is fact (BadAmmoUsed) true
⊞ [1]	Is fact (OKAmmoAvailable) true
⊞ [2]	NOT(Is fact (GoodAmmoAvailable) true)
⊟ ThenAction	...
⊞ [0]	Retract fact (BadAmmoUsed)
⊞ [1]	This is so every instance of using Bad ammo will result in a decrement
⊞ [2]	Modify knowledge level of CorrectAmmoTypeExtended by -0.0500
⊞ [3]	But because we retracted the fact, we need to manually modify the related concept
Weight	**-0.050000**
TickDelay	10
Priority	0
bIsGameCritical	False
⊞ [3]	If Is fact (BadAmmoUsed) true AND Is fact (GoodAmmoAvailable) true Then do 4 actions

Expert System Advantages

- System independent of game
- Expert System Shells
  - Java : Jesse
  - C : Clips
  - Python : Pychinko
- Lots of existing literature and research

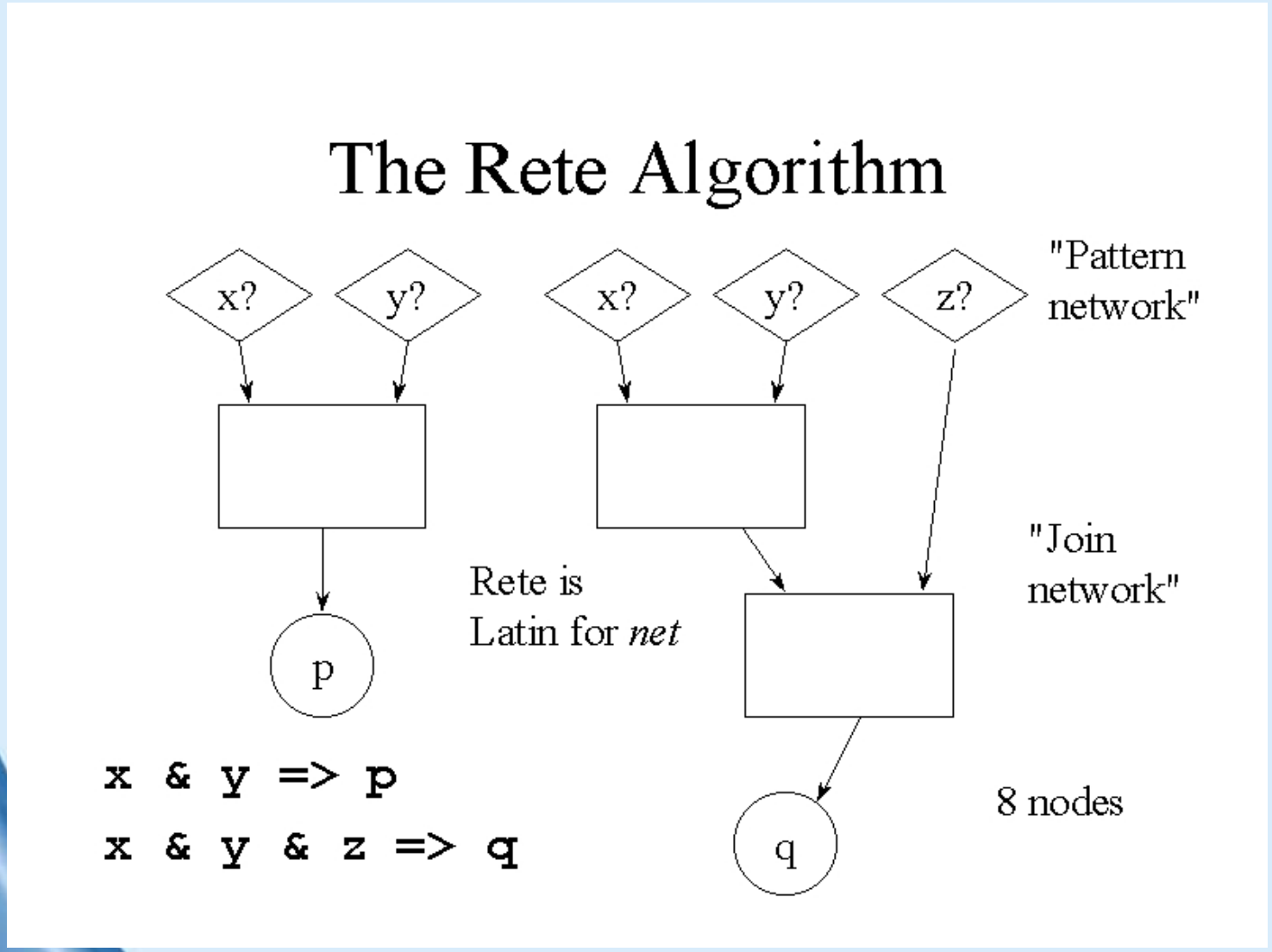
Rete Algorithm

- Avoids linear increase in performance as rules grow
- Latin for ‘Network’
- Converts IF conditions into a data flow network.
- Presents simplified algorithm

Example

- Two rules
  - if x and y Then p
  - if x and y and z Then q
- Evaluates x and y twice
  - Operations could be expensive
  - (?x, Married, ?z) could match a lot of items

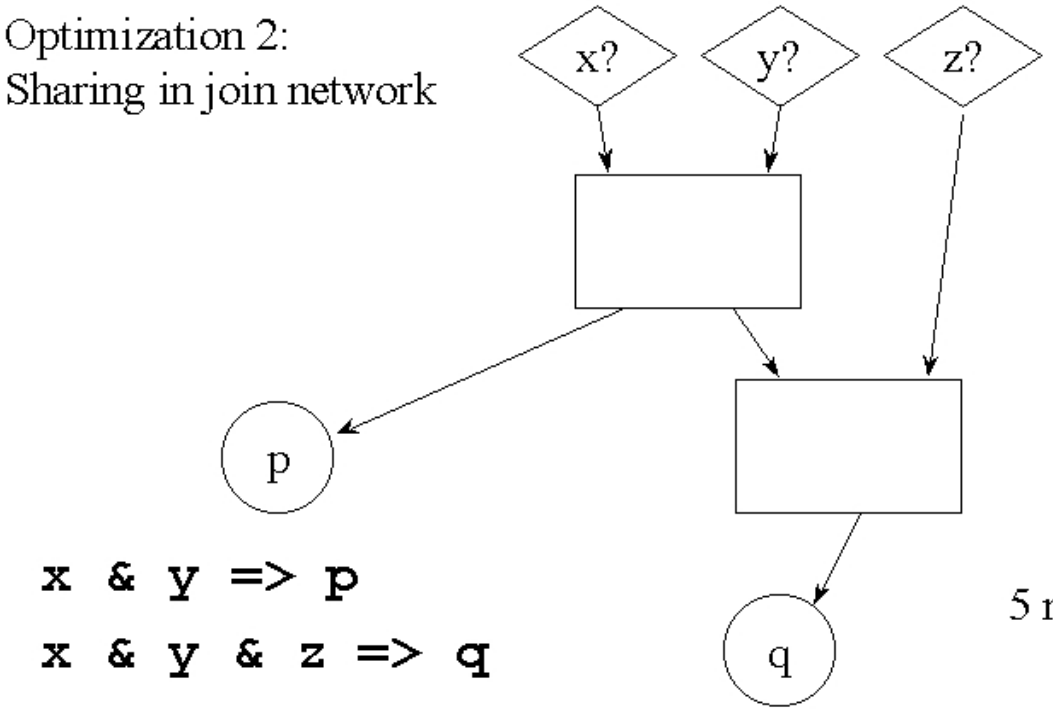
Convert to nodes



Optimize Network

The Rete Algorithm

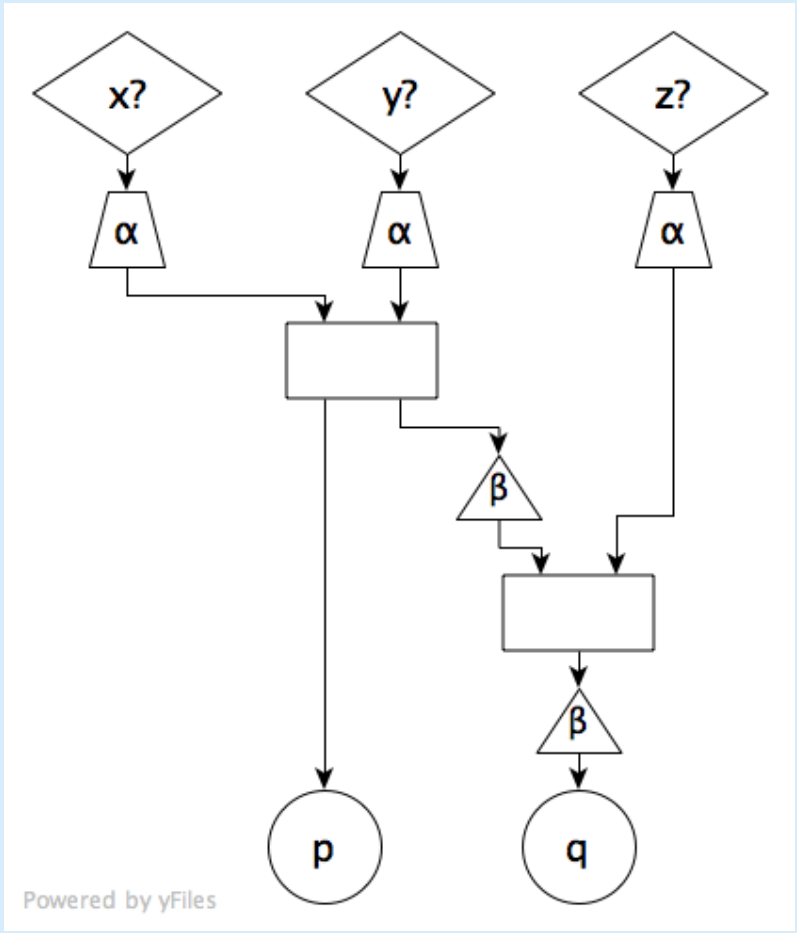
Optimization 2:
Sharing in join network




```
x & y => p
x & y & z => q
```

5 nodes

Alpha/Beta Memory



- Alpha Memory
  - Store all facts that matched pattern
- Beta Memory
  - Stores pairs matched by join nodes
- Only incur cost when facts change
  - Insert or remove from alpha/beta memory

Does Adaptive Training Work?

- Don’t Know Yet
  - Focus testers have found them useful
- Play Bioshock and get back to me.

Future Improvements

- Integration with difficulty system
- Give player situations to facilitate learning.

References

- Rete Paper : Production Matching for Large Learning Systems
- Expert System Shell : CLIPS
- Infinity Script Unofficial Guide