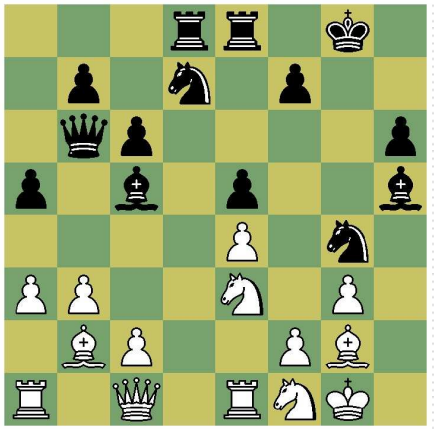
ORTS: A Case Study of Multi-Tasking in Soar

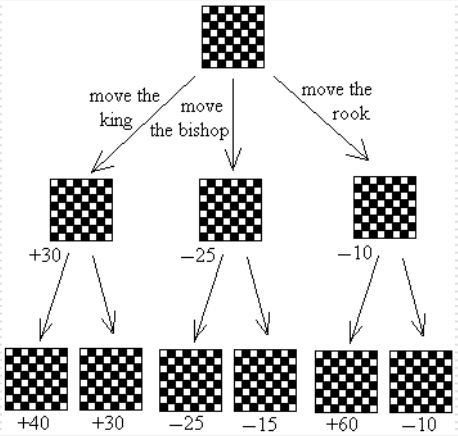
Joseph Xu
University of Michigan
Soar Workshop 2006

Outline

1. Motivation for studying Real Time Strategy games
2. ORTS and AIIDE competition
3. SORTS Design
4. Multi-tasking in SORTS
5. Progress & Conclusions

Traditional Games in AI





- Discrete time/actions
- Perfect information
  - Enumerable states
- Low perceptual/motor load
- Examples:
  - Chess
  - Towers of Hanoi
  - Water Jug

Real-Time Strategy Games

- Played in real time
- Maintain an economy
- Develop production capabilities via cities/bases
- Defend against enemy attacks
- Launch attacks against enemies
- Examples:
  - Starcraft
  - Command & Conquer

Real-Time Strategy Games

- Continuous time/space/actions
- Imperfect information
  - State space not practically enumerable
- High perceptual/motor load

Challenges

	Which game stresses this more?	What's better at this (presently)?
Look-ahead	Chess	Computer
Opponent modeling	Chess	Human
State abstraction	RTS	Human
Spatial/Temporal reasoning	RTS	Human
Manage perceptual overloading	RTS	Human
Multi-faceted gameplay	RTS	Computer
Divided Attention	RTS	Computer

Challenges

	Which game stresses this more?	What's better at this (presently)?
Look-ahead	Chess	Computer
Opponent modeling	Chess	Human
State abstraction	RTS	Human
Spatial/Temporal reasoning	RTS	Human
Manage perceptual overloading	RTS	Human
Multi-faceted gameplay	RTS	Computer
Divided Attention	RTS	Computer

Challenges

	Which game stresses this more?	What's better at this (presently)?
Look-ahead	Chess	Computer
Opponent modeling	Chess	Human
State abstraction	RTS	Human
Spatial/Temporal reasoning	RTS	Human
Manage perceptual overloading	RTS	Human
Multi-faceted gameplay	RTS	Computer
Divided Attention	RTS	Computer

Outline

1. Motivation for studying Real Time Strategy games
2. ORTS and AIIDE competition
3. SORTS Design
4. Multi-tasking in SORTS
5. Progress & Conclusions

Open Real Time Strategy

- Open source RTS implementation
- Designed specifically for AI research
- Completely customizable via scripts
- C++ API
  - receive information about state of the world from server
  - Send commands to server
- Under active development at University of Alberta

AI Competition at AIIDE 06

- Game 1 – Resource gathering
- Game 2 – Offense and Defense
- Game 3 – Full RTS game

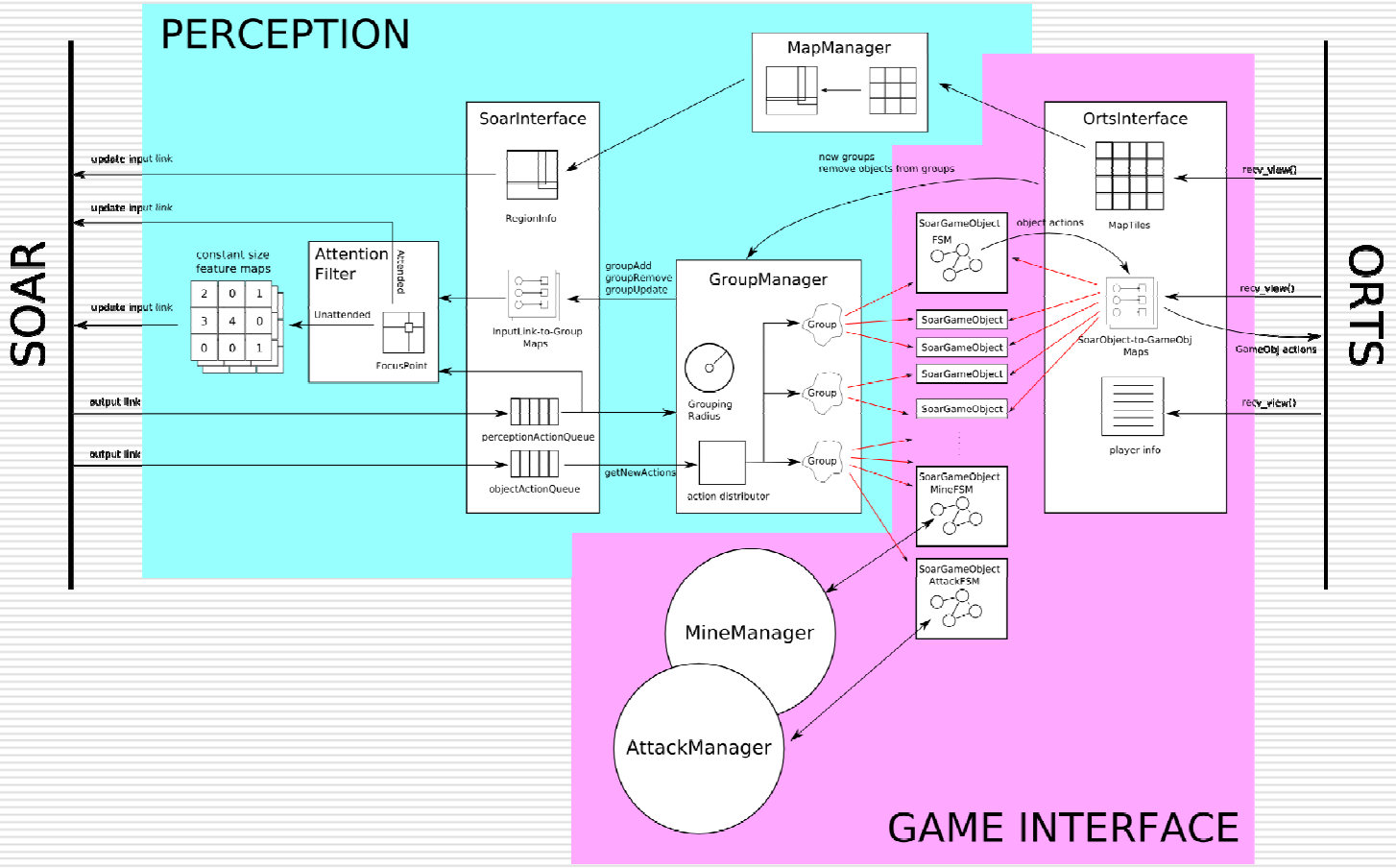
Outline

1. Motivation for studying Real Time Strategy games
2. ORTS and AIIDE competition
3. SORTS Design
4. Multi-tasking in SORTS
5. Progress & Conclusions

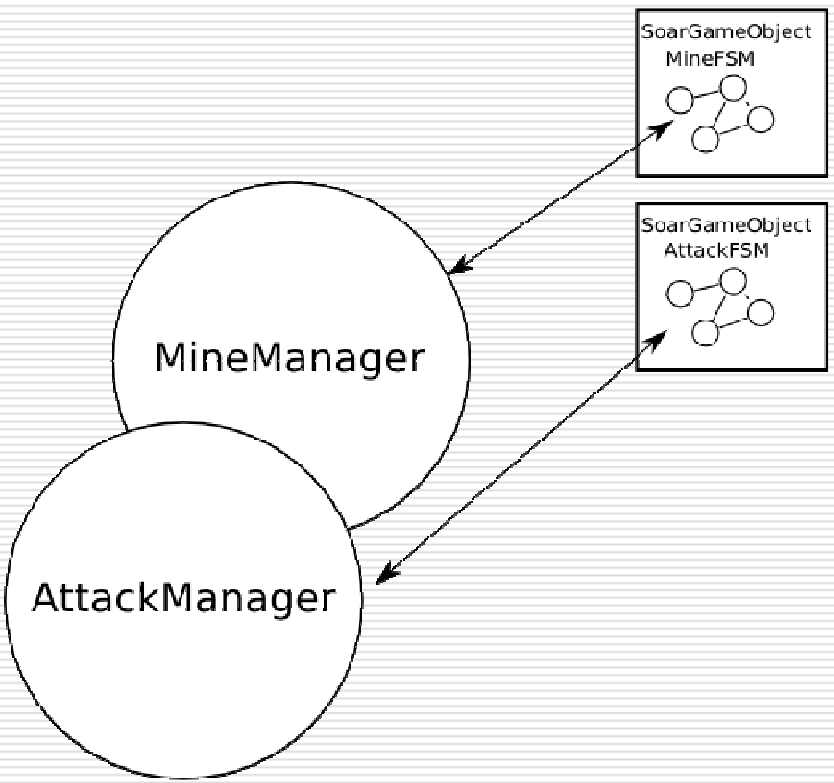
Our Approach: SORTS

- Create a Soar agent to play ORTS
- Middleware serves as both Soar's perceptual system as well as a gaming interface
  - Like a real game interface, the middleware handles micromanagement such as pathfinding and default unit behaviors

SORTS Architecture



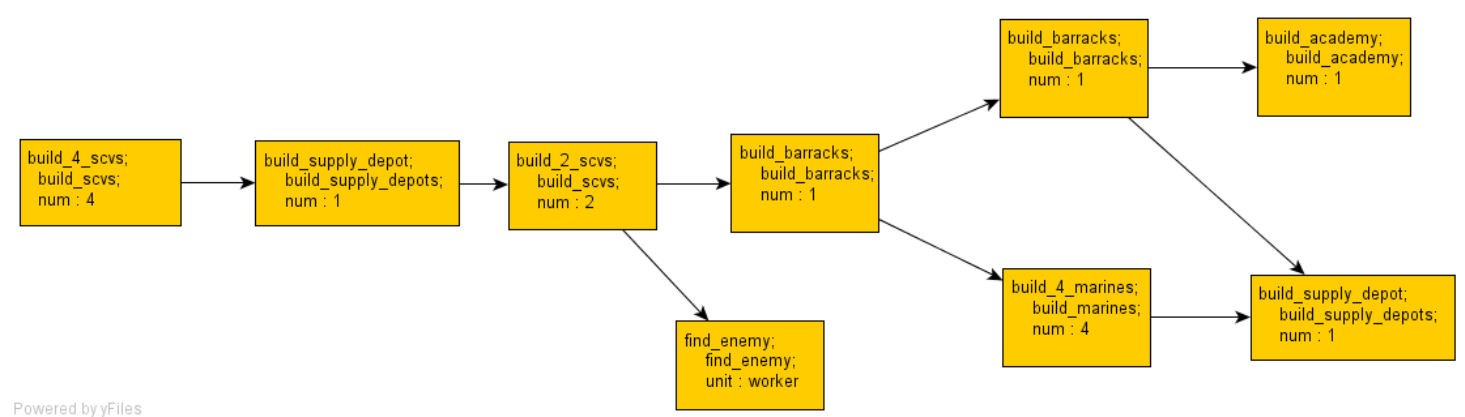
Low Level Control



- Each unit controlled by finite state machines
- Soar agent doesn't have to micromanage
- MineManager and AttackManager necessary for finer control in competition

Soar Agent Preliminary Design

- Three ways of acting
  - Static plans
    - Defined ahead of time, like the opening book in chess
    - Partial Order Plans



Soar Agent Preliminary Design

- Three ways of acting
  - Opportunistic plans
    - Plans that the agent comes up with while playing the game
    - Backward chaining
  - Example: I need to build anti-air defenses to counter enemy fighters, but to do that I need to build a factory first

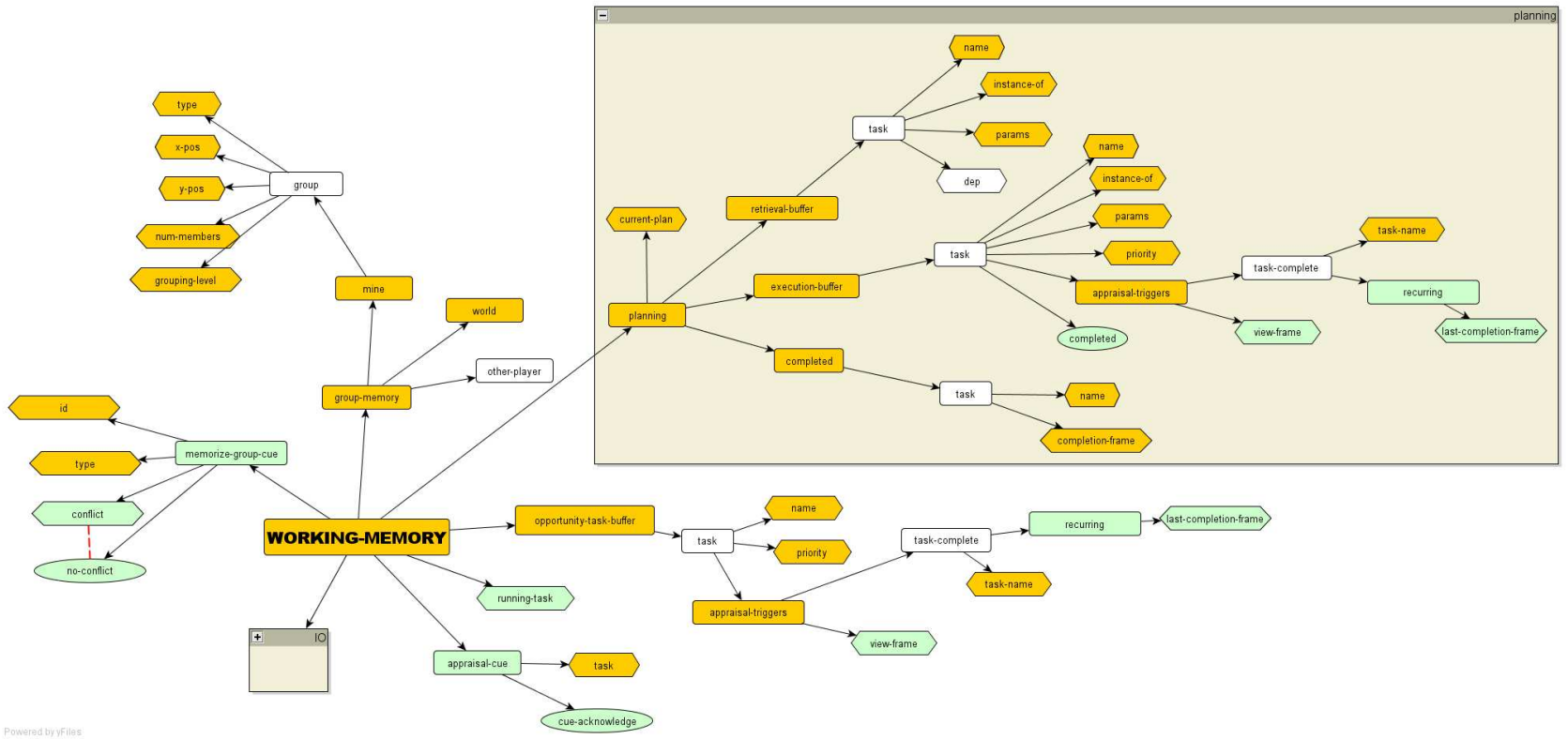
Soar Agent Preliminary Design

- Three ways of acting
  - Reactions
    - Reactions to the current state that can occur at any time
  - Example: I can't win this battle, retreat with remaining forces

Soar Agent Preliminary Design

- Situation Awareness
  - Soar agent can only “see” a limited area of game field at any time
  - Agent must make decisions that account for unseen parts of the game field too
  - Must maintain situation awareness by memorizing important things going on at different parts of the map

Soar Agent Preliminary Design



Division of Responsibilities

- Soar handles
  - State abstraction
  - Planning
  - High level commands
  - Multi-tasking
- MW handles
  - Visual abstraction
  - Command implementation
  - Default unit behaviors
  - "Uninteresting" strategies
    - Mining
    - Micromanage attacks

Outline

1. Motivation for studying Real Time Strategy games
2. ORTS and AIIDE competition
3. SORTS Design
4. Multi-tasking in SORTS
5. Progress & Conclusions

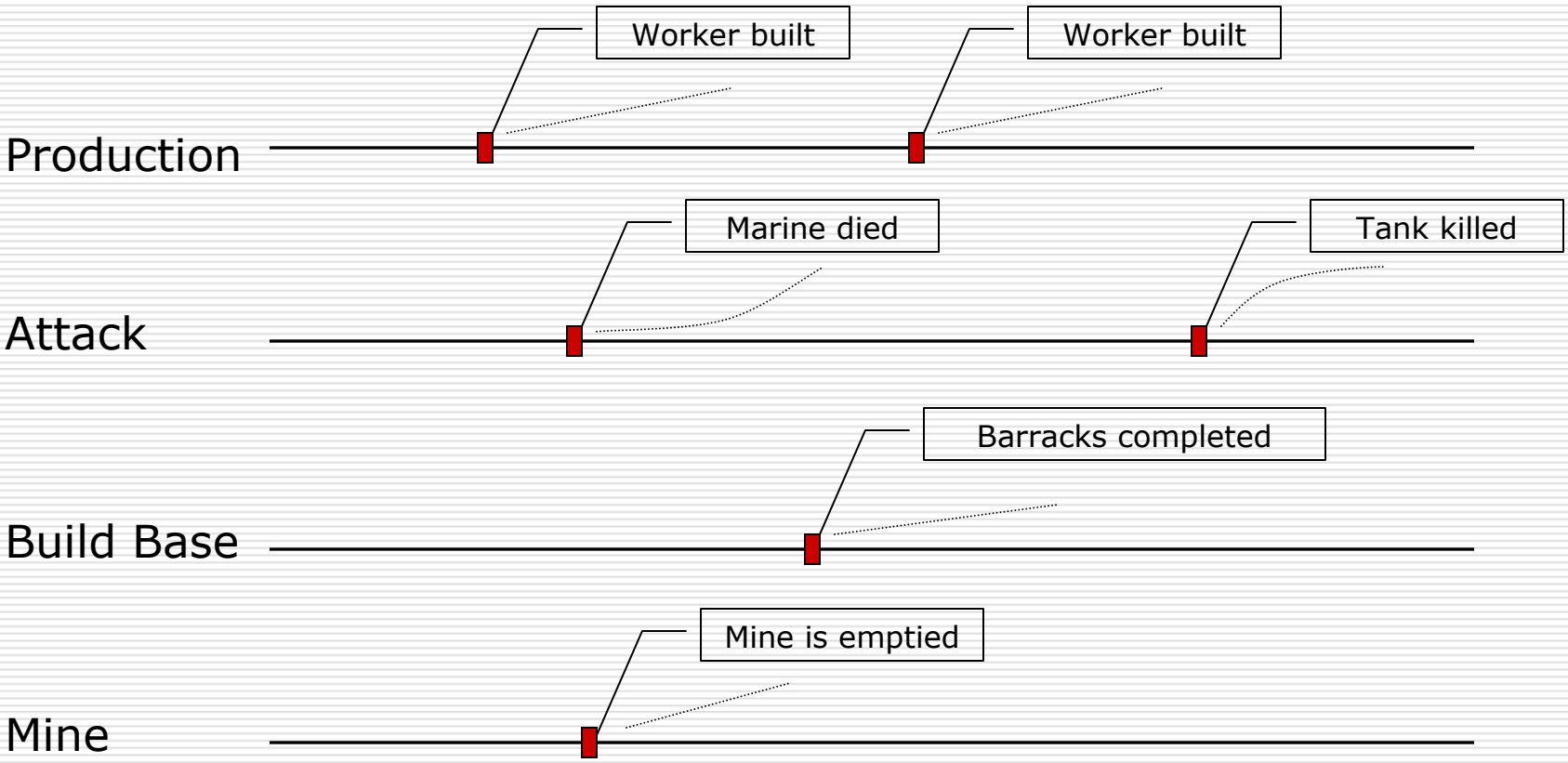
Multi-tasking

- RTS games typically require player to manage many simultaneous tasks
  - Tasks are attentionally and cognitively far apart
    - Hence there is a cost in switching between tasks
  - Attention tunneling is usually detrimental

Channel Model Approach

- Group actions taken over the entire game into tasks
  - How to group is not yet finalized
  - Follow human tendency
- Each task is a *channel*
  - The agent performs some action on the channel then waits for feedback in the form of *Events*
  - Arrival times of events are not known
  - There is a cost incurred that is a function of the amount of time an event remains on a channel unprocessed

Channel Model



Task Switching

- The agent builds an internal model of expected event arrival times on each channel
- Accuracy of the model distinguishes novice and expert players
- Environmental cues help to determine when to switch
- Top-down and bottom-up control

Outline

1. Motivation for studying Real Time Strategy games
2. ORTS and AIIDE competition
3. SORTS Design
4. Multi-tasking in SORTS
5. Progress & Conclusions

Progress

- What is implemented
  - Perceptual system
  - Command system (Game Interface)
  - Low level FSMs
  - Planning
- What has to be implemented
  - Real Soar agents
  - Situation awareness

Progress

- Competition
  - Game 1 – 80%
  - Game 2 – 30%
  - Game 3 – 30%

Conclusions

- RTS games present a set of challenges to AI research that chess does not
- We are building SORTS to try to meet some of these challenges
- SORTS will make a good platform on which to test the new Soar architecture

Conclusions

- Nuggets
  - Provides a rich environment to test many of Soar's new capabilities
  - Forced us to confront issues that would not have come up in other environments / architectures
- Coals
  - Still in pre-alpha stage
  - Some decisions were made in the interest of competition performance rather than psychological plausibility
  - Abuses working memory