Ebonsoar: Toward a Simulated Human Opponent

Tim Hoffman

21st Soar Workshop
May 6, 2001

Introduction

- EBONSOAR: investigations into an alternative game playing framework (FORR).
- Generalize FORR to a broader class of games.
- Compare FORR to the Top-down approach to agent design.

FORR

- A plausible model of human game playing [Epstein 1994].
  - avoids deep search.
  - combines simple heuristics.
- Employs a system of heuristics called *advisors.*
  - Each advisor represents a specific heuristic.
  - Advisors are separated into levels of order and authority, called *tiers*.

Advisor Tiers

- Tier 1
  - Authoritative advisors.
  - Polled in a specific order.
  - A recommended action is performed immediately.
  - No further deliberation is done.

Advisor Tiers (cont'd)

- Tier 1.5
  - Considered only if Tier 1 makes no decision.
  - Authoritative advisors.
  - Polled in a specific order.
  - Advisors perform weak search to arrive at recommendations.
  - Recommended action performed immediately.
  - No further deliberation is done.

Advisor Tiers (cont'd)

- Tier 2
  - Considered only if Tier 1.5 makes no decision.
  - Non-authoritative advisors.
  - Advisors recommend an action with a degree of confidence.
  - A voting scheme decides which action is recommended by the most advisors.

Extending FORR

- FORR was developed for discrete, perfect-information, turn-based games.
- Question: is FORR extensible to real-time, continuous, interactive games?
  - convert continuous variables to ‘buckets’ of values.
  - shallow searches involving partitioned continuous variables may quickly become intractable.

The Subtleties of Time

- When time is partitioned, different advisors can recommend actions in different time frames.
- Actions may require a certain amount of time to complete themselves.
  - A lower tier advisor could override a higher tier's action while being performed.
  - When time is partitioned, different advisors can recommend actions in different time frames.
- Requires *Persistent Justifications*.

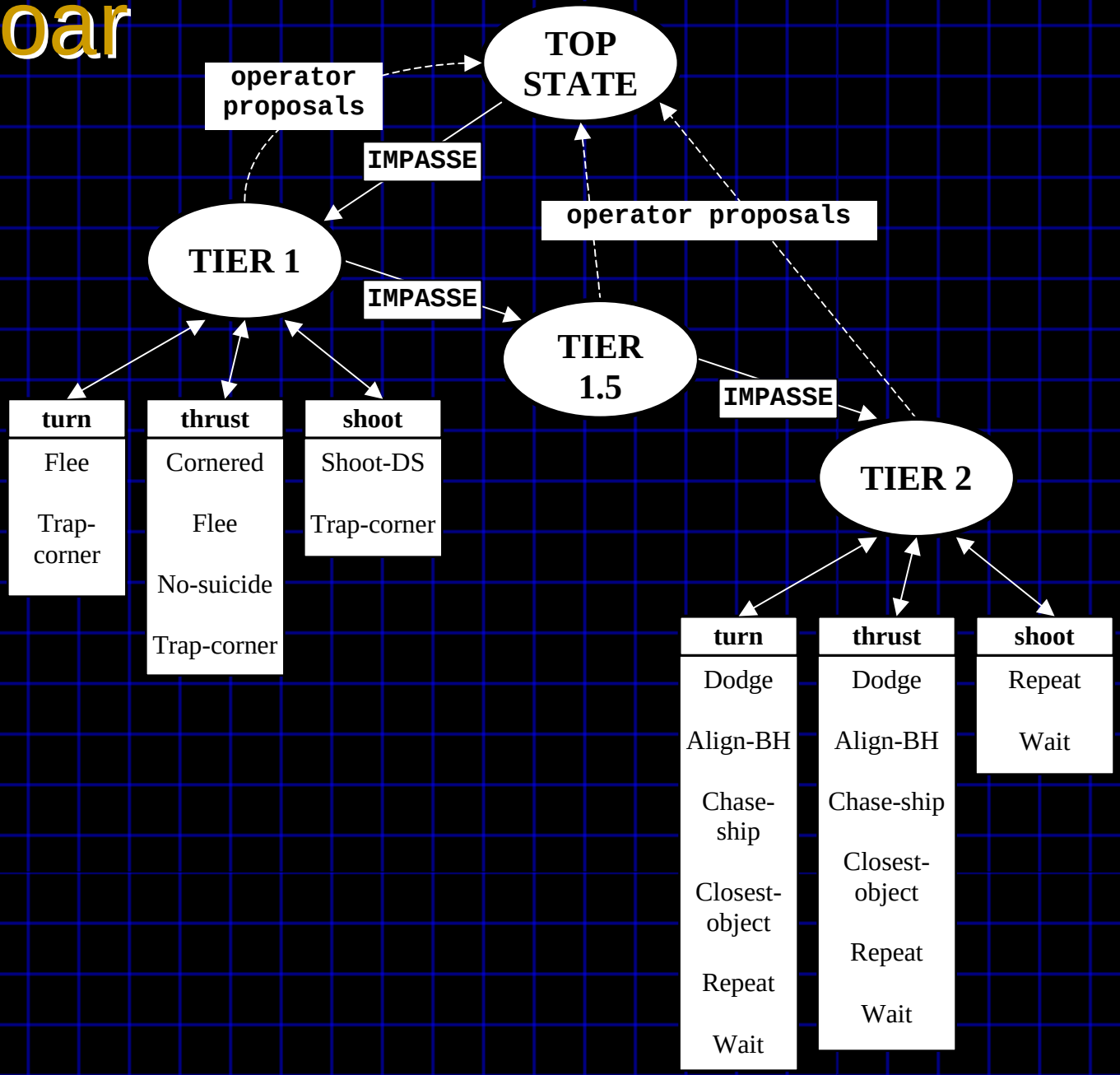
Persistent Justifications

- Store the preconditions for an action when an advisor recommends it:
  - Action name
  - Advisor rank
  - Conditions 1,2, ... n that justify the action
- Persistent Justifications remain in working memory until one or more conditions become false.
- When an advisor recommends an action, it must check to see if a justification exists for that action.
  - If the justification contains an advisor whose rank is equal to or greater than than the advisor recommending the action, then no action is taken.

Implementation

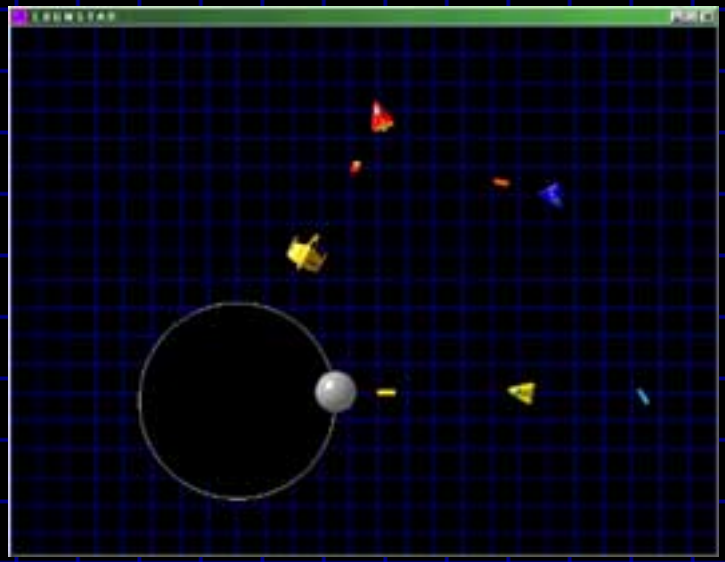
- Created EBONSOAR, an Asteroids-type arcade game, as a testing environment.
- Interfaced EBONSOAR and Soar with SGIO.
- Designed a FORR framework in Soar.
- Developed a set of basic heuristics to serve as advisors.

FORR in Soar


TOP STATE
operator proposals
IMPASSE
TIER 1
IMPASSE
operator proposals
TIER 1.5
IMPASSE
TIER 2
turn
Flee
Trap-corner
thrust
Cornered
Flee
No-suicide
Trap-corner
shoot
Shoot-DS
Trap-corner
turn
Dodge
Align-BH
Chase-ship
Closest-object
Repeat
Wait
thrust
Dodge
Align-BH
Chase-ship
Closest-object
Repeat
Wait
shoot
Repeat
Wait

EBONSOAR Architecture

EBONSOAR



SGIO

GAME STATE

AGENT ACTIONS

SoarAgent

C++

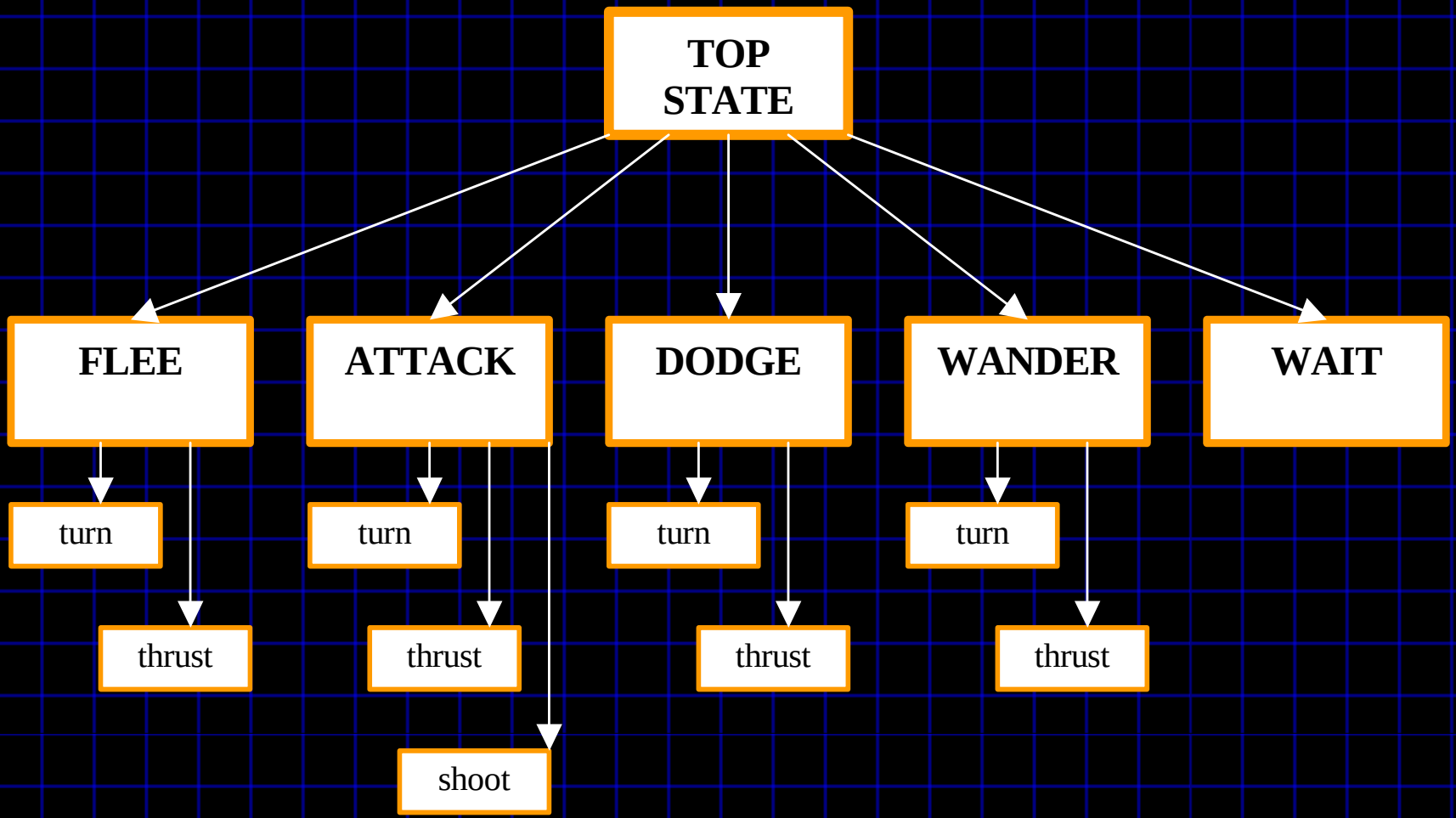
INPUT / OUTPUT LINKS

Advisors in **Soar**

AGENT ACTIONS

GAME STATE

Top-down hierarchy



FORR vs. Top-down

- FORR exhibited thrashing.
  - Differing, alternating majorities in Tier 2.
  - Weighting the votes of advisors can help.
  - Developer must provide sufficiently many Tier 2 advisors for voting.
- Advisors in FORR are “plug & play”.
- Top-down ran smoothly (no thrashing.)
- Top-down is potentially more complex if the number of subgoals becomes large.

Nuggets & Coal

- Nuggets:
  - FORR can be extended to more complex domains.
  - Advisors easily added to and removed from the framework.
  - Side effect: SoarAgent class can be reused.

- Coal:
  - Thrashing.
  - Careful tweaking of Tier 2 voting weights required to avoid unrealistic behavior.
  - Learning component in FORR not addressed yet