Athena Analyst's Guide: Politics

Athena S&RO Simulation, V3

May, 2011

William H. Duquette

[William.H.Duquette@jpl.nasa.gov](mailto:William.H.Duquette@jpl.nasa.gov)

Jet Propulsion Laboratory

*© 2008-2011. California Institute of Technology. ALL RIGHTS RESERVED.*

*U.S. Government sponsorship acknowledged. Any commercial use must be negotiated with*

*the Office of Technology Transfer at the California Institute of Technology.*

*The technical data in this document may be controlled under the*

*U.S. Export Regulations, release to foreign persons may require an export authorization.*

*This document has NOT been reviewed for export sensitive content.*

Table of Contents

1. Introduction 4

2. Politics 5

2.1 Definition of Terms 5

2.2 Actors 5

2.3 Goals 6

2.3.1 Possible Extensions 6

2.4 Tactics 7

2.4.1 Actions 8

2.4.2 Cost Functions 8

2.5 Conditions 9

2.6 Tactic Execution Algorithm 9

2.6.1 A Road Not Taken 10

3. Catalog of Tactics and Conditions 12

3.1 Tactic Types 12

3.2 Condition Types 13

# Introduction

This is a working document tracking the design and implementation of the Athena Politics model in Athena 3.0. Sections 2 and 3 of this document is intended for eventual inclusion in the *Athena Analyst's Guide*. The goal of this document is to describe the elements of the Politics model as they are implemented in version 3.0, to serve as a foundation for further development. As such, the emphasis is more on what the model is, and less on why it is what it is how it relates to the real world.

This version of the document is intended to describe actors, goals, tactics, and conditions. Other concepts in the Political model (such as control of a neighborhood, the support relationships, and so forth) are ignored.

# Politics

The Athena Politics model describes the role of significant political actors in the Athena playbox. Each actor has goals, and in order to reach those goals they make use of a variety of tactics. Tactics can change the state of affairs in the playbox. Every tactic has a cost in money and personnel, and might have a number of attached conditions governing when the actor will choose to make use of it.

## Definition of Terms

An **actor** is an individual or group of individuals that functions as a significant decision-maker in the politics of the playbox, e.g., the president of Pakistan, the leadership of Al Qaeda, or the United States Government. The relevant actors in a scenario depend on the scope of the scenario. If the playbox consists of a single city, the mayor of the city is likely to be a significant actor; if the playbox consists of an entire country, mayors of individual cities are likely to be irrelevant.

Every actor can have zero or more **goals**. A goal is a state of affairs that the actor wishes to bring about through his actions. For example, the actor might wish to take control of a particular neighborhood. A goal consists of a set of Boolean predicates called **conditions**; the goal is set to be **met** if all of the conditions are true, and **unmet** otherwise.

The actor's strategy for reaching his goals is specified as a prioritized list of **tactics**. Each tactic represents an action the actor can take. Tactics require the use of assets (money and personnel), and as these are scarce the actor cannot do everything they might want to do.

The actor might have any number of **conditions** attached to a given tactic. The actor will consider executing a tactic only if all of the attached conditions are true. For example, most tactics will have an attached condition specifying the goals that the tactic supports. The tactic will be considered for execution only if at least one of the listed goals is unmet.

The meanings of these terms will be fleshed out in the following sections.

## Actors

An actor is an individual or group of individuals that functions as a significant decision-maker within the playbox. Each actor *a* has the following attributes:

|  |  |
| --- | --- |
| **Name** | **Description** |
| income | Income in dollars per tactics tock (nominally seven days) |
| cash\_on\_hand | Cash on hand, in dollars: the money the actor has immediately available to fund tactics. Unspent cash\_on\_hand is carried over to the next tock. |
| cash\_reserve | Cash reserve, in dollars. The actor can reserve funds for later use using the SAVE tactic, and release them for spending later on using the SPEND tactic. |

In addition, each actor can own force or organization groups, which it can use to carry out its tactics. Presence of personnel belonging to these groups can also contribute to the actor's control of a neighborhood.

## Goals

A goal is a state of affairs that an actor wishes to bring about. The goal is defined by a set of Boolean predicates called **conditions**; the goal is set to be **met** if all of the conditions are true, and **unmet** otherwise. Each goal has the following data attributes:

|  |  |
| --- | --- |
| **Name** | **Description** |
| *owner* | The ID of the actor to whom the goal belongs. |
| *state* | The state of the goal: "normal" or "disabled". The actor will pursue goals that are both enabled and unmet. The analyst will be able to disable goals temporarily. |

### Possible Extensions

This section describes a number of possible extensions to the model that we may pursue at a later time.

**Progress Metrics:** A progress metric is a measure of the actor's progress toward a goal, probably expressed as a percentage. This would be nice to have; however, I suspect that defining and implementing progress metrics for the conditions that define a goal might be more difficult than it seems.

**Activation Conditions:** As defined above, the actor pursues goals that are both enabled and unmet, and the analyst controls the *state* flag. It might be desirable to allow activation conditions to be attached to goals just as they are to tactics; the goal would be enabled only if all activation conditions were met. For example, Goal G1 might be enabled if:

* Goal G2 has been met.
* Goal G3 has *not* been met.
* The simulation time is less than six months after time 0.

This would allow the actor to pursue chains of goals, and to decide to pursue one goal in preference to another.

However, adding conditions like these to goals can cause the condition evaluation network to have cycles, which would have to be detected and prevented.

**Actions on Success/Failure:** A goal succeeds when its condition is met; Ed Upchurch has suggested that a goal fails if an end-time is reached before the goal succeeds. Certain actions could be taken on goal success or failure, such as activating another goal. Note that some of the actions Ed suggests would also be handled by "Activation Conditions", above.

## Tactics

A tactic is an action the user can choose to take, possibly in support of one or more goals. Athena defines a number of kinds of tactic; these are referred to as **tactic types**. The complete list is documented in Section 3.1 . Each tactic type has the following data attributes:

|  |  |
| --- | --- |
| **Name** | **Description** |
| *owner* | The ID of the actor to whom the tactic belongs. |
| *priority* | An actor's tactics are listed in priority order from 1 to *N*; the tactic's *priority* is its position in the list. |
| *narrative* | A human-readable string that summarizes the action of the tactic. |
| *state* | "normal", "disabled", or "invalid". "Normal" tactics are eligible for execution. The analyst can disable a tactic, and a tactic can be marked "invalid" if it becomes invalid after it has been created, e.g., if a required force group is no longer owned by the tactic's owning actor. |
| *exec\_flag* | 1 if the tactic was selected for execution at the last tactics tock, and 0 otherwise. |
| *exec\_ts* | The simulation time, in ticks, of the last tactics tock at which the tactic was selected for execution. |
| type-specific parameters | These vary from tactic type to tactic type. For example, in the tactic "Group *g* defends with ROE *roe* in Neighborhood *n*", *g*, *roe,* and *n* are the type-specific parameters. |

In addition, every tactic has the following operations:

|  |  |
| --- | --- |
| **Name** | **Description** |
| *narrative* | Computes the tactic's narrative text. |
| *check* | Sanity-checks the tactic's parameters against the current state of the simulation. |
| *dollars* | Returns the tactic's estimated cost in dollars, given its parameters and the current state of the simulation. The actual cost at execution may be different. |
| *execute* | Attempts to execute the tactic, returning true on success and false on failure. The tactic must first determine if there are sufficient resources, possibly scaling its effect to the resources available; if there are, it then expends those resources and executes its effect. |

In addition, each tactic type has a *cost function* and an *action*, both of which depend on the type-specific parameters.

### Actions

A tactic is how an actor makes a change in the playbox; thus, every tactic has some associated *action*, the set of changes it makes. Actions vary with the tactic type; see Section 3.1 for details.

### Cost Functions

Taking action has a cost; and the actions an actor can take are therefore constrained by the actor's resources. Athena models two resources: personnel and money. Personnel are drawn from the force and organization groups owned by the actor; money is simply money. For the purposes of this design, we assume that every actor has a weekly income[[1]](#footnote-1) and a stock of money called the actor's **cash-on-hand--**the cash the actor has available to fund tactics.

The tactic's cost function, then, computes the cost of the tactic, in dollars and personnel, from the tactic's parameters. If 300 troops are assigned to man checkpoints in a neighborhood, for example, the tactic will cost 300 personnel and some number of dollars per person. The number of dollars might depend on the group from which the personnel come, the nature of the neighborhood, and the nature of the checkpoints.

The cost of a tactic can be estimated as soon as the tactic is created, and the estimated cost should be immediately available to the user.

Cost functions vary with the tactic type; see Section 3.1 for details.

## Conditions

By default, tactics are executed in priority order until the actor's available assets are used up. (See Section 2.6 for the algorithm.) However, the user can attach any number of *conditions* to each tactic. Each condition is a Boolean predicate; the tactic will be selected for execution only if all attached conditions are true. For example, an actor might use a particular tactic only if he is in control of a particular neighborhood, or only if the cost is a small fraction of his available resources. Most especially, the actor might use a particular tactic only if a particular goal or set of goals is unmet.

Every condition has the following set of data attributes:

|  |  |
| --- | --- |
| **Name** | **Description** |
| *tactic\_id* | The ID of the specific tactic to which the condition is attached. |
| *narrative* | A human-readable string, summarizing the condition and its parameters. |
| *state* | "normal", "disabled", and "invalid", as for tactics. Only conditions in the "normal" state are queried as part of tactic selection. |
| type-specific parameters | These vary from condition type to condition type. For example, in the condition "Actor is in control of Neighborhood *n*", *n* is the type-specific parameter. |

Every condition has the following set of operations:

|  |  |
| --- | --- |
| **Name** | **Description** |
| *narrative* | Computes the condition's narrative text. |
| *eval* | Evaluates the condition, returning true or false. The condition's truth value may depends on its parameters and on the current state of the simulation. |

See Section 3.2 for the catalog of condition types.

## Tactic Execution Algorithm

The actor's prioritized list of tactics represents his strategy for achieving his goals. At each tactic execution interval (nominally one week)[[2]](#footnote-2), the following algorithm is used:

Evaluate all goal and condition flags.

For each actor *a:*

Determine the list of eligible tactics, i.e., those for which all dependent conditions are met.

For each actor *a*:

For each of *a*'s eligible tactics *T*, in priority order:

Attempt to execute *T*:

If insufficient resources are available:

Fail; continue with next tactic.

Expend resources.

Execute tactic.

### A Road Not Taken

We also considered a different approach:

* Actor *a*'s goals are prioritized from most important to least important.
* Every tactic is associated explicitly with one or more goal.
* The tactics associated with a goal are in priority order for that goal.

Then, the algorithm is as follows:

For each actor *a:*

Let the *plan* be the empty list.

For each of *a'*s goals *G*, in priority order,

For each tactic *T* associated with *G*, in priority order:

If *T* is already in the *plan*, skip it.

If *T*'s conditions are not met, skip it.

If *T*'s cost exceeds the available resources, skip it.

Reduce *a*'s resources by *T'*s cost.

Add *T* to *a*'s *plan*.

For each tactic *T* in *a'*s *plan*,

Execute tactic *T*.

This is a very natural approach, when you start with goals and then move on to the tactics required to achieve them. Unfortunately, it yields unrealistic results because it assumes that each goal in the list dominates all subsequent goals: the actor will do everything possible to meet the first goal before moving onto the second, and so on. In reality, an actor might wish to push forward towards several goals at once, using the most important tactic for each, before moving on to less important (or less effective) tactics.

By prioritizing the tactics instead of the goals, and attaching "Goal Unmet" conditions to them, we allow the analyst to decide which tactics the actor is likely to use in support of his goals, and to order them to reflect the actor's likely very complex priorities—instead of trying to write a model to do that, and getting it wrong.

# Catalog of Tactics and Conditions

This section will contain a catalog of the tactics and conditions defined by Athena. This catalog is TBD. Note that this catalog (or some form of it) should probably move to the as yet unwritten User's Guide.

## Tactic Types

My prototype code defines the following tactics. Note that actor *a* is an implicit input to each tactic.

|  |  |
| --- | --- |
| **DEFROE(*g,n,roe*): Set Defensive ROE** | |
| **Narrative:** | Group *g* defends in neighborhood *n* with ROE *roe.* |
| **Constraints:** | Group *g* must be a uniformed force group owned by actor *a*. |
| **$/week** | n/a |
| **Personnel:** | 0. This tactic determines a group's ROE, not its activity. It simply applies to all of *g*'s personnel in *n*. |
| **Action:** | Sets group *g*'s ROE in the neighborhood. This tactic replaces the existing interface to specify defensive ROEs for uniformed force groups. |

|  |  |
| --- | --- |
| **DEMOB(*g, mode, personnel, once*): Demobilize Forces** | |
| **Narrative:** | Demobilize *personnel* of group *g'*s available personnel. |
| **Constraints:** | Group *g* must be a force group owned by actor *a*. |
| **$/week** | 0 |
| **Personnel:** | *personnel* |
| **Action:** | Removes personnel from the playbox. The personnel must be available for deployment. If *mode* is **SOME**, then the tactic removes up to *personnel* available personnel; if the *mode* is **ALL**, then the tactic removes all available personnel.  If *once* is true, then the tactic will execute only once; its state will be automatically set to "disabled" on execution. It will not execute a second time unless manually re-enabled by the user. |

|  |  |
| --- | --- |
| **DEPLOY(*g, mode, personnel, nlist*): Deploy Forces** | |
| **Narrative:** | Deploy *personnel* of group *g'*s available personnel into neighborhoods *nlist*. |
| **Constraints:** | Group *g* must be a force group owned by actor *a*. |
| **$/week** | *personnel* \* *cost*, where *cost* is group *g*'s maintenance cost in $/person/week. |
| **Personnel:** | 0. This tactic makes personnel available for assignment. |
| **Action:** | Places personnel into neighborhoods so that they can assigned particular duties. If *mode* is **SOME**, then at least *personnel* must be available, and the cost must be affordable, or the tactic will not execute.  If *mode* is **ALL**, then the tactic will attempt to deploy all of the group's remaining undeployed personnel; if there are insufficient funds, the tactic will deploy as many of the troops as the actor can afford. This is intended for use as a catch-all at the low-priority end of the actor's strategy, so that all troops are deployed somewhere; undeployed troops are automatically demobilized. |

|  |  |
| --- | --- |
| **SAVE(*percent*): Save Money** | |
| **Narrative:** | Save *percent*% of income to the actor's cash reserve for later use |
| **Constraints:** | None. |
| **$/week** | , where *income* is the actor's income for this tock. |
| **Personnel:** | 0 |
| **Action:** | Saves a percentage of the actor's income to his cash reserve. If less than the desired amount of cash is unavailable, the tactic will save whatever is left. |

|  |  |
| --- | --- |
| **SPEND(*percent*): Spend Money** | |
| **Narrative:** | Releases *percent*% of the actor's cash reserve for spending on tactics. |
| **Constraints:** | None. |
| **$/week** | , where *reserve* is the actor's cash reserve. The value is negative, because it will be immediately added to the cash available for tactics, instead of subtracted. |
| **Personnel:** | 0 |
| **Action:** | Removes a percentage of the actor's reserve. |

## Condition Types

Conditions are currently documented in the Application Help.

1. The actor's income will eventually be computed by the Economics model. [↑](#footnote-ref-1)
2. WHD: Perhaps this should be one day; and tactics should have a length in days. Tactics persist until their length has been exceeded. Some things happen quickly and others slowly, and some things have great momentum. We will eventually need a way to handle that. [↑](#footnote-ref-2)