Astrological Description Language: A Proposal

Rangarajan Krishnamoorthy
July 25, 2005
ranga@mmsindia.com

This document gives an *informal* description of a *semi-formal* language for describing astrological concepts.

Goal:
To design a language for describing concepts, predications and rules pertaining to astrology. The language must be reasonably simple to use without sacrificing rigour.

Entities:
There are several predefined entities in the language.
1) Planets := {Su, Mo, Ma, Me, Ju, Ve, Sa, Ra, Ke, Ch, Fo}
2) Signs:= {Ar, Ta, Ge, Cn, Le, Vi, Li, Sc, Sg, Cp, Aq, Pi}
3) Houses := {1H, 2H, 3H, 4H, 5H, 6H, 7H, 8H, 9H, 10H, 11H, 12H}
4) Cusps := {1C, 2C, 3C, 4C, 5C, 6C, 7C, 8C, 9C, 10C, 11C, 12C}

As you would have noticed, we can define a collection of entities within braces. This collection is called a "set".

Variables:
Sometimes we will talk about entities without knowing their actual identity. Variables are like placeholders for "real" things. We will denote variables by meaningful names prefixed by a question mark.
Examples: ?planet, ?cusp, ?somehouse

Functions:
Functions apply to one or more entities and return a value.

1) Sgl(?entity): This returns the signlord of the given argument. The argument can be a planet or a cusp.
2) Stl(?entity): This returns the starlord of the given entity.
3) Sl(?entity): This returns the sublord of the given entity. Sublord can be computed to any length. We can use the function Ssl(?entity) to compute the subsublord, Sssl(?entity) to compute the subsubsublord, etc. Another way is to use the function Sbl(?level, ?entity) to compute the sublord of an entity to any given depth. In fact, Sl(?entity) == Sbl(1, ?entity).
4) House(?planet): This returns the house in which the planet is located.
5) Sgn(?entity): This returns the sign in which the cusp or planet is located.
6) Str(?entity): This returns the star in which the cusp or planet is located.
7) Degree(?entity): Returns the zodiac position in egrees of the given entity.

Boolean Functions:
The following functions return true or false.

8) IsPlanet(?p): Returns true if ?p is a planet, else returns false
9) IsCusp(?p): Returns true if ?c is a cusp, else false
10) IsHouse(?h): returns true if ?h is a house else false
11) IsSign(s): Returns true if ?s is a sign,else false
12) IsRetrograde(?p): returns true if the planet ?p is retrograde, else false
13) IsDirect(?p): returns true if the planet ?p is direct, else false.

Aspect Functions:

We have a few functions to talk about both Vedic and Western aspects.

14) VAspect(?planet1, ?planet2): Returns the vedic aspect between the two planets ?planet1 and ?planet2. For example in a particular chart if we find that Mars aspects Mercury by 4^{th} aspect, we will say
    VAspect(Ma, Me) == 4^{th}
15) VAspect(?planet, ?cusp): Vedic aspect between a planet and a cusp.
16) WAspect(?planet1, ?planet2): Western aspect between ?planet1 and ?planet2.
    Example: WAspect(Ma, Sa) == Square

When there is no aspect between two entities, the function will return "Nil". For example if in a chart there is no western aspect between Mo and Ju, then the following will hold good:

WAspect(Mo, Ju) == Nil

Checking *Set* Membership:

We will find use for a function that checks whether an element is part of a given set.

17) Member(?element, ?a_set): Returns true if the element denoted by ?element is in the set defined by ?a_set. Otherwise it will return false.

Logical Operators:

When we wish to describe conditions and rules in our language, we will require the use of special operators that apply to Boolean expressions. These operators are 'and', 'or', 'not'. These can also be written as '&&', '||' and '!' respectively.

The above is just a partial list. We will keep adding to this whenever required. We will definitely require arithmetic operators in our language. Parentheses will come to our aid in controlling operator precedence.

Now instead of spending too much time on the language definition part, let us see how our language can be used.

Usage Examples:

1) Let us define a new concept called "connection" between two planets.

Connected(?planet1, ?planet2) ⇔
    Sgl(?planet1) == ?planet2 || Stl(?planet1) == ?planet2
        || Sl(?planet1) == ?planet2
        || Sgn(?planet1) == Sgn(?planet2)
        || WAspect(?planet1, ?planet2) != Nil

The above definition states that a planet A is connected to a planet B if and only if *any* of the following is true:

a) A is in the sign of B,
b) A is in the star of B,
c) A is in the sub of B,
d) A is in the same sign as B,
e) There is *some* western aspect between A and B.

2) Udupa's example (in his posting): "Shukra is posited in 2nd House in Nakshatra Lord Guru, in Sublord Budha" can be expressed thus:

House(Ve) == 2 && Stl(Ve) == Ju && Sl(Ve) == Me

3) There is a YOD configuration in a chart when three planets have a particular aspect relationship.

YOD(?planet1, ?planet2, ?planet3) ⇔
WAspect(?planet1, ?planet2) == Quincunx &&
WAspect(?planet1, ?planet3) == Quincunx &&
WAspect(?planet2, ?planet3) == Sextile

4) A native has the potential to become an astrologer when

a) The asc sublord is connected with Sa and Me, and
b) The asc sublord is connected with both 9 and 12 houses

Define Rule Astrologer *:=*
Connected(Sl(1C), Sa) && Connected(Sl(1C), Me)
&& Connected(Sl(1C), 9H)
&& Connected(Sl(1C), 12H) ➔
Assert("Native has the potential to become an astrologer")

I have not defined *Connected(?planet, ?house*) relationship. However, you could define that to complete the above rule.

Conclusion: I have sketched the structure of ADL as I would like it to be. This is by no means final or complete. Members are invited to share their comments.

*** END ***