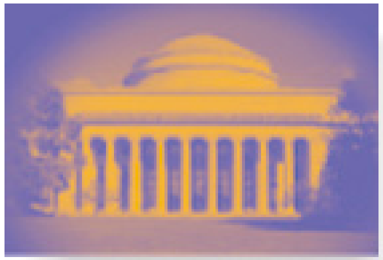
NKS 2004

A New Kind of Language for Complex Engineering Systems:

Case Study: NASA's Apollo Program



Benjamin Koo
Edward F. Crawley
Engineering Systems Division
Department of Aeronautical and Astronautical Engineering

Architecting in Engineering Systems

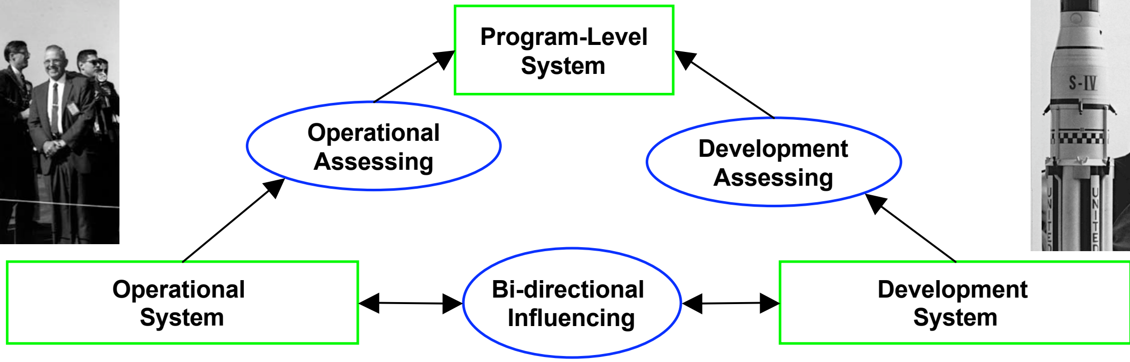
- One Grand Challenge in Engineering Systems:
  - Assessing ***Very Large Scale*** engineering decisions early enough with meaningful model resolutions and abstraction levels
  - Architects face numerous options based on possible combinations and permutations with insufficient computational power
- Need to integrate multiple domains of knowledge at meaningful levels of abstraction and resolution
  - By applying the Principle of Computational Equivalence to ***translate between domain models***
  - Using a generalized version of Wolfram Automata: ***Object-Process Network (OPN)***
- Applying NKS to NASA's Apollo Mission:
  - Wolfram's automata (a simple language) can be mapped onto multiple abstraction levels based on the ***Principle of Computational Equivalence***
  - ***NASA's Apollo mission*** is revisited to to show that NKS can help assess critical decisions at multiple technical levels

Architecting as Language Manipulation

- The Principle of Computational Equivalence states that all physical processes can be mapped onto equivalent "languages"
  - Computer scientists often define various kinds of "machines" as "languages" or "automata"
- All domain-specific models must be built on a common set of linguistic primitives in order to ensure consistency
  - Choice of vocabulary embodied in ***OBJECTS*** with their respective range of admissible states define the variability in a system
  - Grammatical rules are ***PROCESSES*** that capture the relationships between objects (as conditional probability functions) can reduce system complexity
  - Vocabulary arranged by Grammatical Rules creates a ***NETWORK***

Language: the medium of interactions

Language(s) of Decision Makers



Language(s) of Operators

Language(s) of Technologists

- Interactions between subsystems requires translation between domain-specific languages
- System architects must share a ***global language*** or ***translate*** between local/domain-specific languages
- All domain-specific models must be built on ***a common set of linguistic primitives*** in order to ensure consistency and composability

Languages based on “Simple Programs”

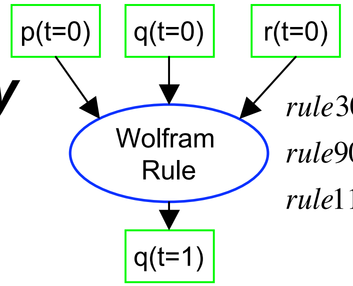
- Wolfram Automata
  - A series of interactive machines that follow ***one*** rule to perform Turing computable tasks
- Languages based on a “simple kernel”
  - Mathematica
    - Multi-Paradigm Programming Language
    - Meta-Language that subsumes multiple programming paradigms
  - Modelica
    - A visual object-oriented language for physical systems based on Mathematica’s runtime engine

Variability in Cell-State Permutation

- Cellular Automata is a visual programming language

```
<cells, neighboring relations, initial condition, rule number>
```

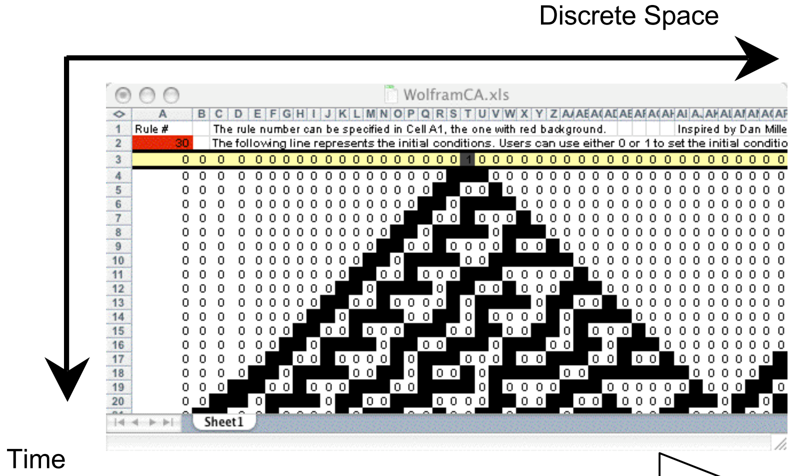
- Order sensitive: color permutation at initial row
- The vertical dimension is ***time***
- A ***common vocabulary*** and ***one rule family***



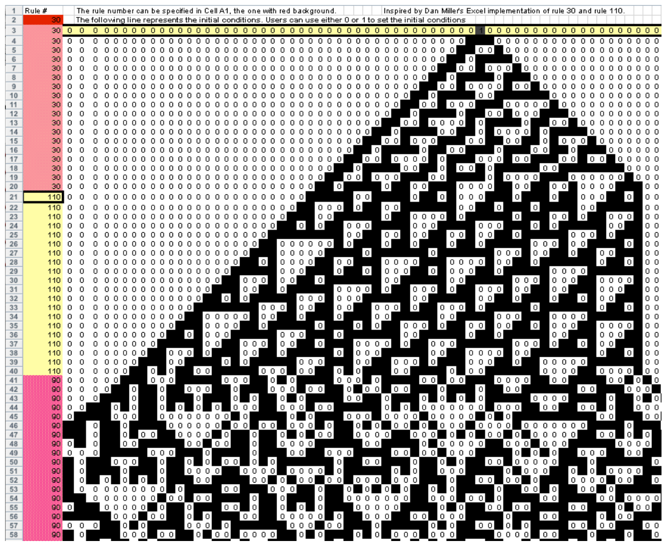
$rule30 : p \otimes (q \vee r)$

$rule90 : p \otimes r$

$rule110 : ((\neg p) \wedge q \wedge r) \otimes q \otimes r$



Wolfram rules are highly composable



Simple Programs in Conditional Probability Tables

- Wolfram Rules can also be encoded in Conditional Probability Tables

$$\Pr(black \mid p,q,r,N=30) = \begin{cases} 0, \{pqr\} \in \{\blacksquare\blacksquare\blacksquare, \blacksquare\blacksquare\square, \blacksquare\square\blacksquare, \square\square\square\} \\ 1, \{pqr\} \in \{\blacksquare\square\square, \square\blacksquare\blacksquare, \square\blacksquare\square, \square\square\blacksquare\} \end{cases}$$

$$\Pr(white) = 1 - \Pr(black)$$

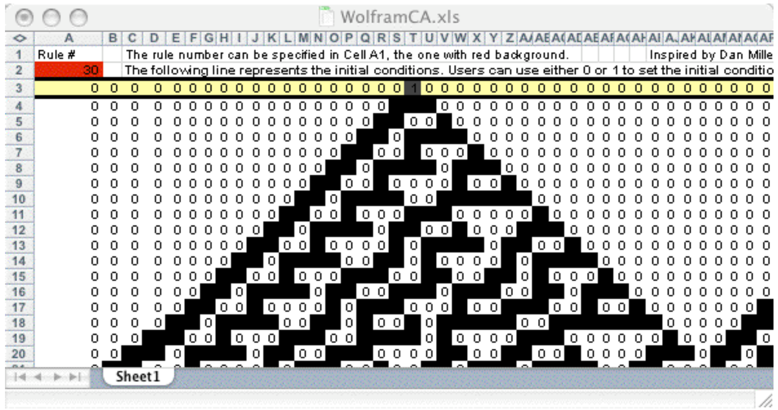
```
Where:
N: rule number (0..255)
p: left cell
q: middle cell
r: right cell
```

One generalized function for all 256 rules… (one alternative to NKS p.648)

$$N \in \{0,1,2...\}$$

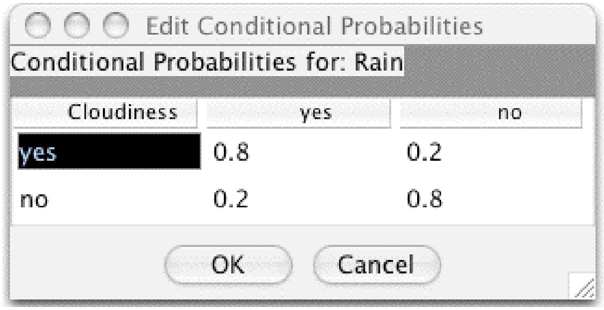
$$c = p \times 4 + q \times 2 + r$$

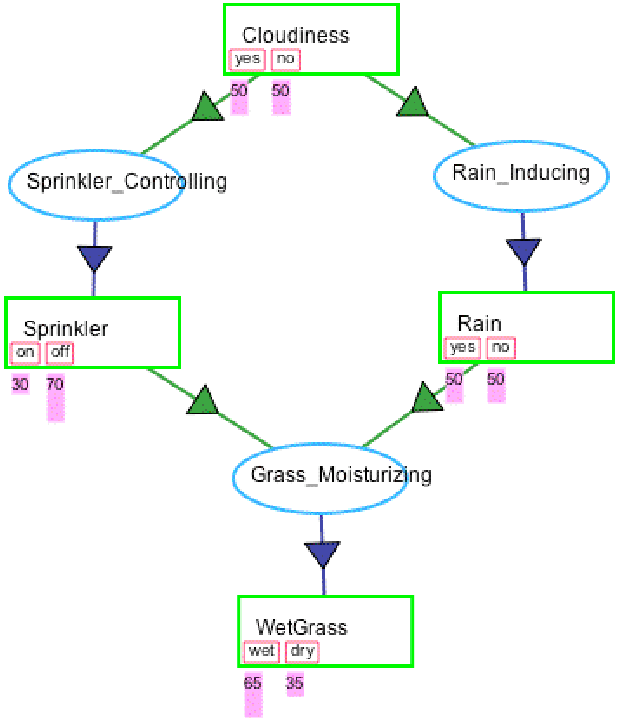
$$\Pr(black, N) = \mathrm{mod}\left(\frac{N - \mathrm{mod}(N, 2^C)}{2^C}, 2\right)$$



Variability in Cell-State Combination

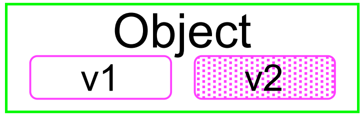
- When ***time is ignored***, a combinatorial problem can be formulated as a graph of conditional probability tables, a.k.a. ***Bayesian Belief Networks (BBNs)***
- BBNs can be thought of as ***probabilistic automata***
- Exact BBN solution is known to be NP-hard, not suitable for large graphs or dynamic problems



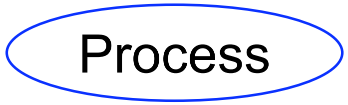


OPN is a Simple Meta-language

- ***OPN*** allows non-technical users to construct domain-specific languages
  - Domain-specific Vocabulary (***Objects***)



    - Local knowledge confines the space of ***combinatorial*** possibility by carefully choosing the inclusion of variables and their admissible value ranges
  - Domain-specific Grammar (***Processes***)

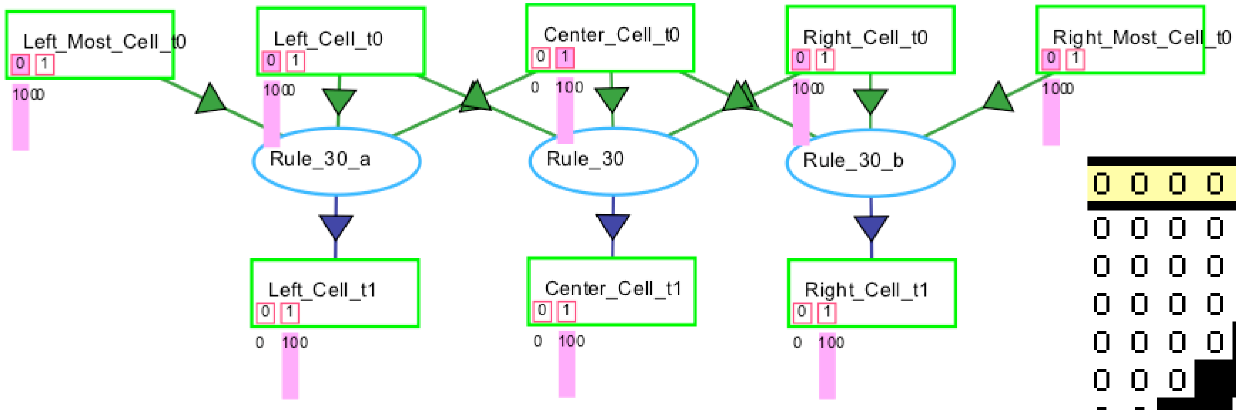


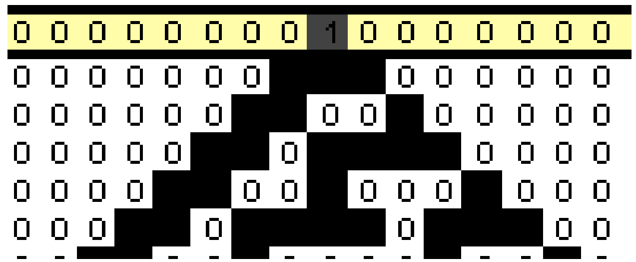
    - Local knowledge eliminates unnecessary ***permutation*** and ***combinatorial*** possibilities
  - Domain-specific language (***Network***)

    - Humans and machines can ***incrementally edit*** the network structure or the conditional probability table based on local context and runtime experience

Cellular Automata in Object-Process Network

- Object-Process Network (OPN) is an extension from Dori's (2002) Object-Process Methodology (OPM)
  - A general purpose system description language (UML replacer)
- Cells as Objects
  - Each object can have two or more states
- Rules as Functions (Processes)
  - Each rule is specified by a unique instance of Conditional Probability Table/Function



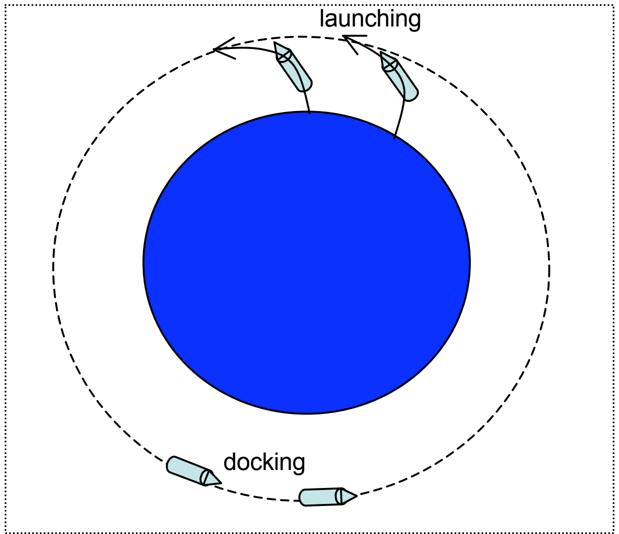


OPN subsumes 4 languages

Languages / Concepts	System Dynamics	Coloured Petri-Net	Probabilistic Network	Cellular Automata	Object-Process Network
Operand	Stock	Place Place + Coloured tokens	State		Object v1 v2
Operator	flow	Transition	p		Process
Relationship	Embedded in arrows	Pre-condition P1 T1 P2 Post-condition	Conditional Probability Tables	Rule number, Cell Layout	Direction of Dependency
Runtime Engine	Numerical Integration Engine	Event Scheduling Engine	Belief Propagation Algorithms	Synchronous Rule Firing	S-Expression Interpreter
Application Domains	Physical System Modeling/Decision Support	Discrete Event Modeling	Reasoning about State-Space Combination	Natural Science Pattern Generation	System Description/ Simulation
Temporal Scale	Infinitesimal Time Steps	Asynchronous Time Steps	Time-independent Memory-less	Synchronized Time Steps	Approximation of multiple scales
Semantic Metaphors	Dynamics of Analog Signals	Dynamics of Messages	Causal Structures	Interactive Systems	Symbolic Knowledge

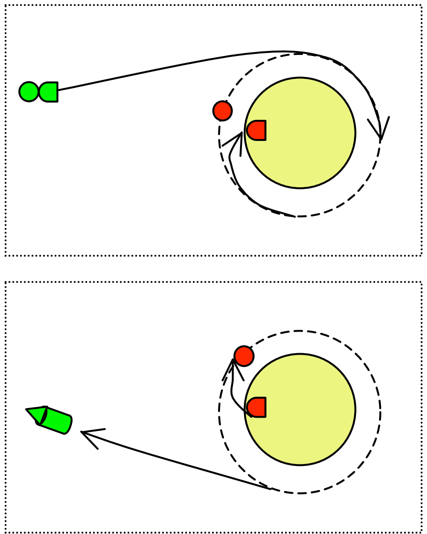
Can one language help reason-through a ***binary decision*** in the ***Apollo Program***?

- A highly public architectural decision
  - Tremendous impact on downstream developmental activities
- How did the decision makers reason through the decision with incrementally available knowledge?



Earth Orbit Rendezvous (EOR)

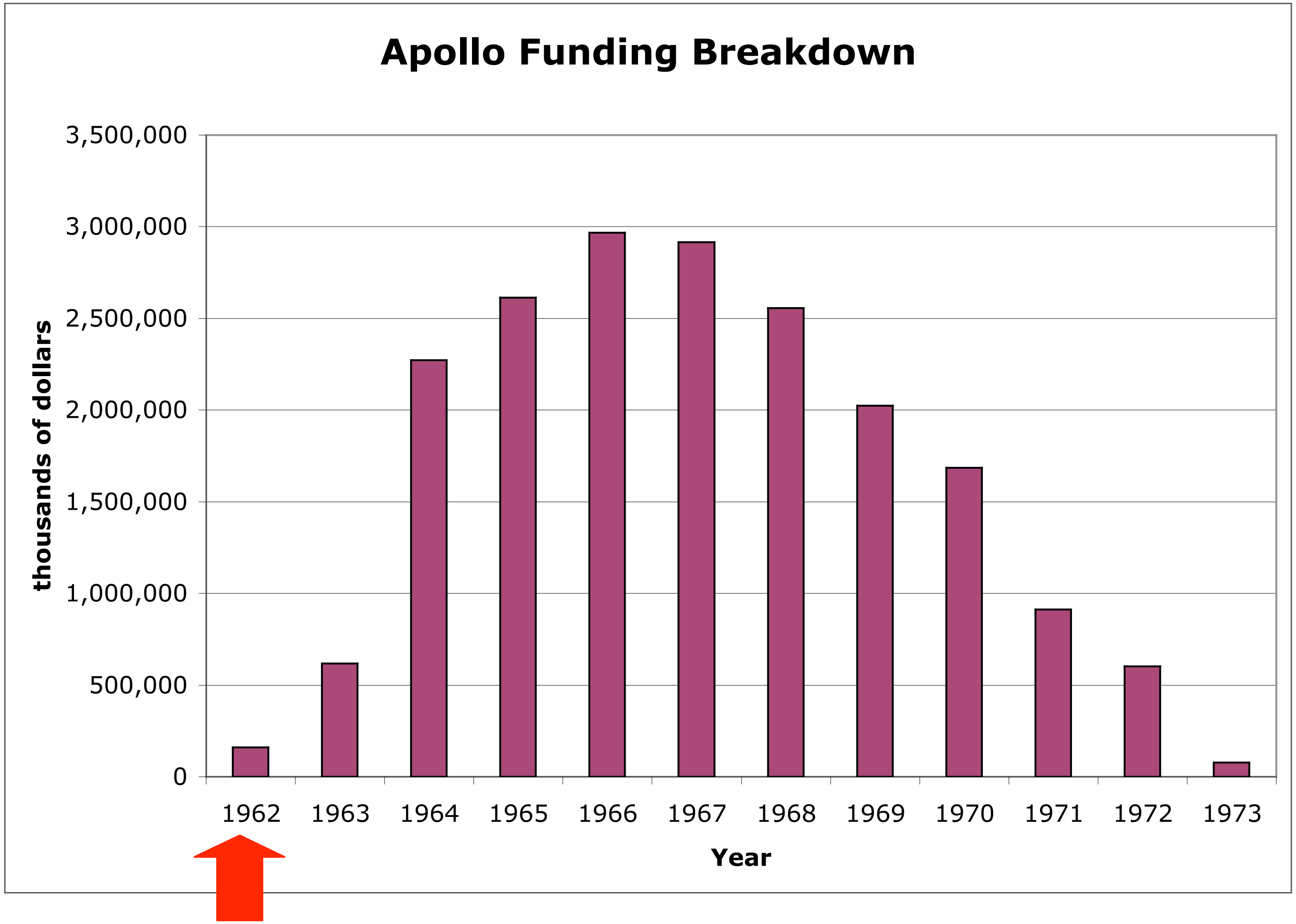
or



?

Lunar Orbit Rendezvous (LOR)

Impact of LOR Architectural Decision



LOR decision reached: June 7th, 1962

$160M, 0.82% of total budget $19.4B

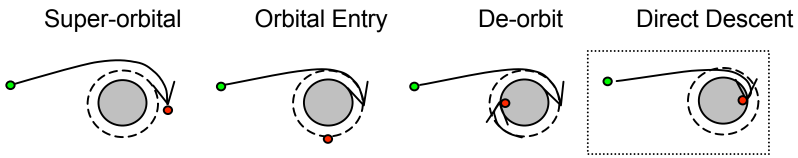
Science in the LOR vs. EOR Decision: Sequence and Combination Matters

- How to enumerate all possible itineraries?
  - Space of trajectories must include ***sequence*** info.
- How to assess variable interactions over multiple knowledge domains?
  - Space of technical options includes a large ***combination*** of possibilities
- How to inform stakeholders about decisions with comprehensive contextual data?
  - The interactions between the two kinds of computational complexity, ***sequence*** and ***combination***, must be coherently organized in a unified representational scheme, namely a ***language***

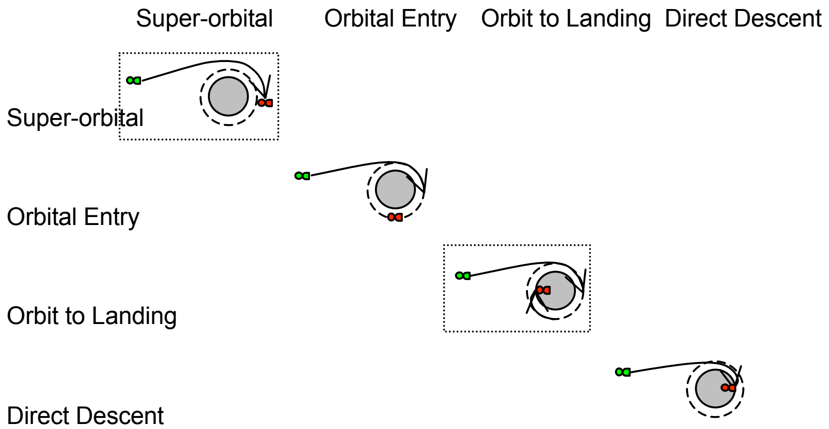
Manual representation and enumeration of trajectory options?

- Consider the trip as four planetary encounters (Earth depart, moon arrive, etc)
- Each encounter has numerous permutations of objects and processes

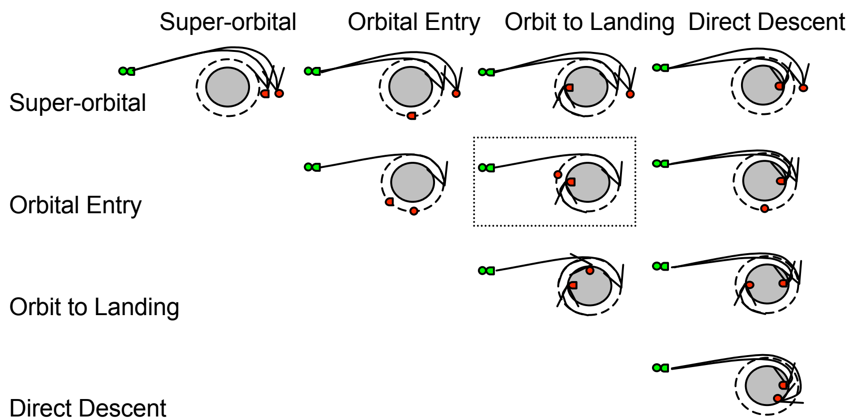
Planetary Arrival - 1 craft



Planetary Arrival - 2 Craft - Joined Initially, Joined Finally

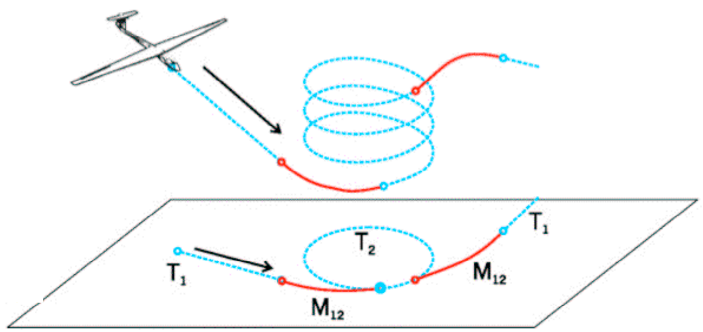


Planetary Arrival - 2 Craft - Joined Initially, Separated Finally



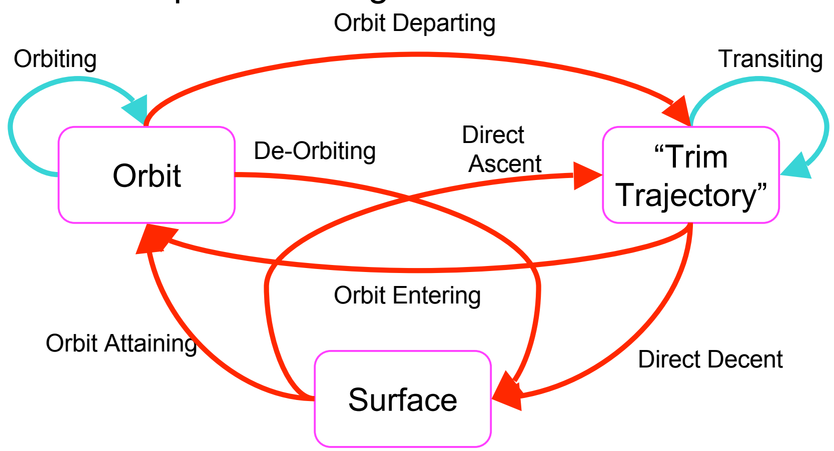
How to represent and enumerate trajectory options systematically?

- Use a ***finite state automaton*** to generate all possible mission architectures
  - Repeatable Motion Primitives (“Trim” conditions)
    - Constant in control setting, configuration
  - Finite-Time Motion Primitives (“Maneuvers”)
    - Finite time transition between two Repeatable Motions

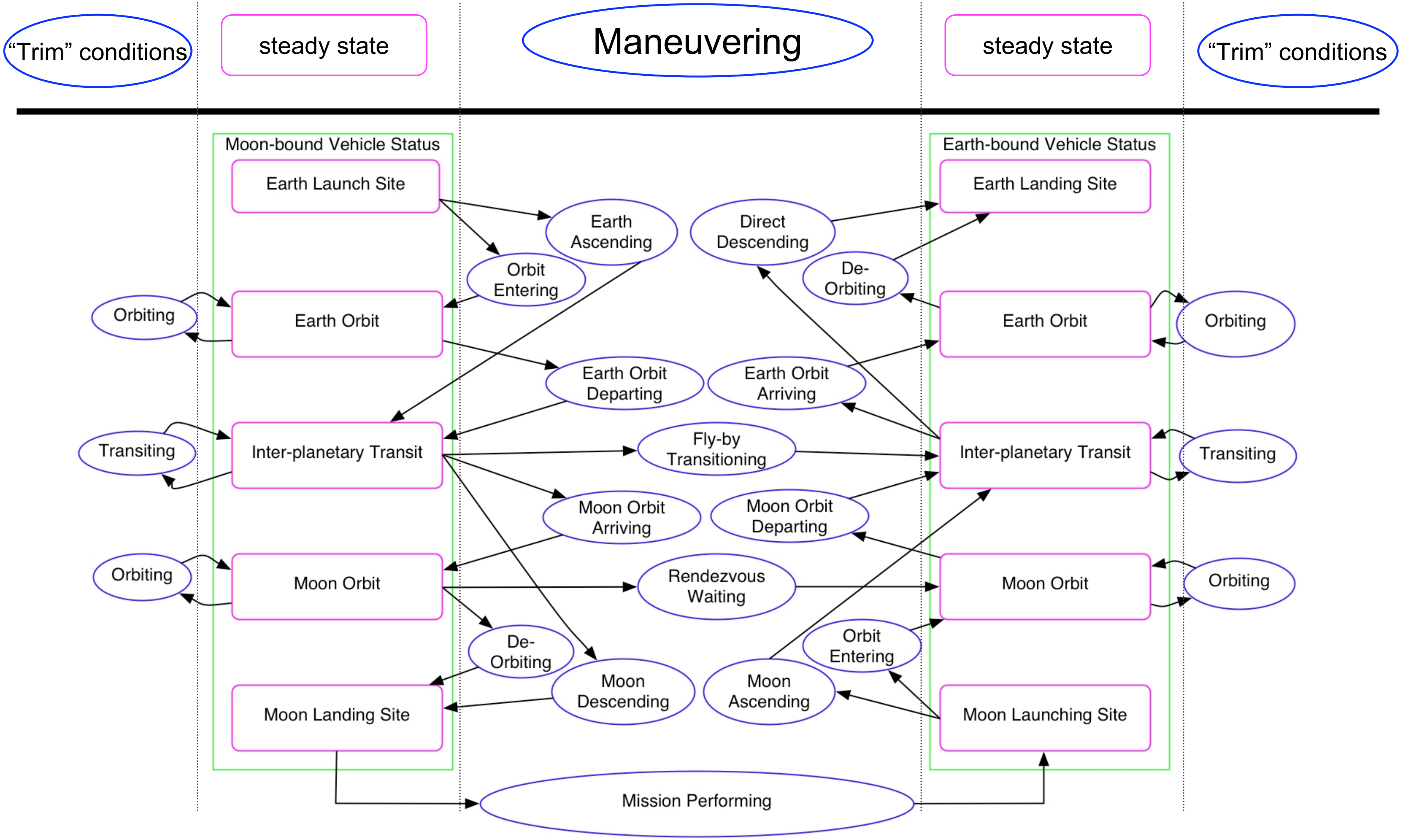


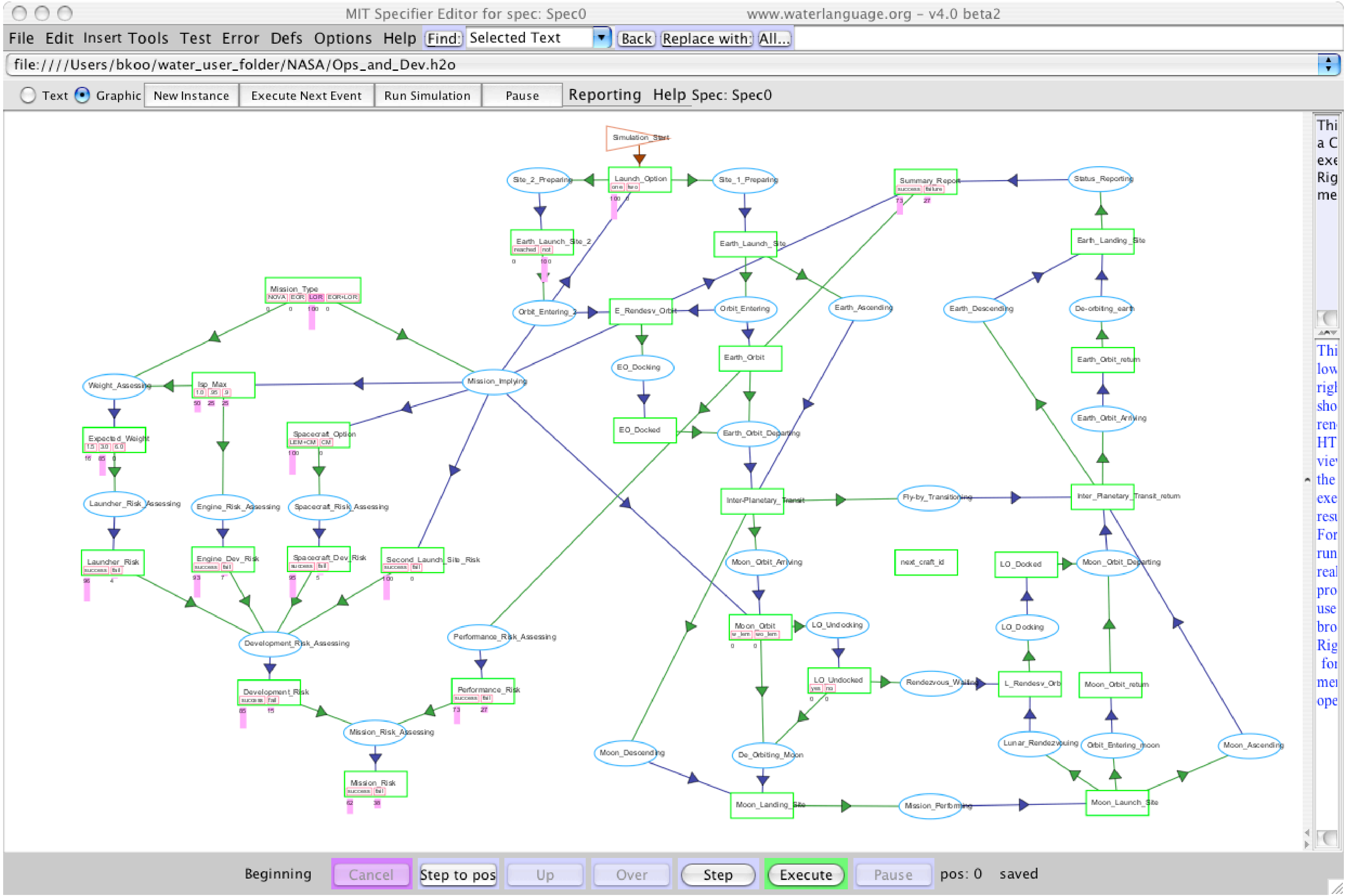
Frazzoli 2001

A “language” for describing all possible single craft itineraries



Generator of Mission Architectures




MIT Specifier Editor for spec: Spec0
www.waterlanguage.org – v4.0 beta2
File Edit Insert Tools Test Error Defs Options Help Find: Selected Text Back Replace with: All...
file:////Users/bkoo/water_user_folder/NASA/Ops_and_Dev.h2o
Text Graphic New Instance Execute Next Event Run Simulation Pause Reporting Help Spec: Spec0
Simulation_Start
Launch_Option
Site_2_Preparing
Site_1_Preparing
Earth_Launch_Site_2
Earth_Launch_Site
Summary_Report
Status_Reporting
Earth_Landing_Site
Mission_Type
NOVA EOR LOR EOR+LOR
Orbit_Entering_2
E_Rendesv_Orbit
Orbit_Entering
Earth_Ascending
Earth_Descending
De-orbiting_earth
EO_Docking
Earth_Orbit
Earth_Orbit_return
Weight_Assessing
Isp_Max
Mission_Implying
Earth_Orbit_Arriving
Expected_Weight
Spacecraft_Option
LEM+CM CM
EO_Docked
Earth_Orbit_Departing
Launcher_Risk_Assessing
Engine_Risk_Assessing
Spacecraft_Risk_Assessing
Inter-Planetary_Transit
Fly-by_Transitioning
Inter_Planetary_Transit_return
Launcher_Risk
Engine_Dev_Risk
Spacecraft_Dev_Risk
Second_Launch_Site_Risk
Moon_Orbit_Arriving
next_craft_id
LO_Docked
Moon_Orbit_Departing
Development_Risk_Assessing
Performance_Risk_Assessing
Moon_Orbit
LO_Undocking
LO_Docking
Development_Risk
Performance_Risk
LO_Undocked
Rendezvous_Waiting
L_Rendesv_Orb
Moon_Orbit_return
Mission_Risk_Assessing
Moon_Descending
De_Orbiting_Moon
Lunar_Rendezvousing
Orbit_Entering_moon
Moon_Ascending
Mission_Risk
Moon_Landing_Site
Mission_Performing
Moon_Launch_Site
Beginning Cancel Step to pos Up Over Step Execute Pause pos: 0 saved

Summary

- Large Scale Engineering Decisions need a New Kind of Language
  - As systems become ***more complex***, ***a simpler yet unifying language*** is needed to deal with the two essential sources of computational complexity (permutation and combination)
  - Existing computable languages can be simplified or emulated using a unifying meta-language, Object-Process Network
- Object-Process Network is a user-friendly meta-language that allows a wide range of users to create ***intuitive***, ***domain-specific***, yet ***efficient*** languages
  - Languages derived from OPN can be composed into a unified computational model to assess the ***interactive effects of subsystems*** across many levels of abstraction and model resolutions

Acknowledgements

- Dr. Robert Seamans for his generous time and oral history that shed light on Apollo's landmark decisions
- Dr. Raymond Leopold for his insights into the design of Iridium Satellite systems and how simple rule tables can be used to construct very complex interactive systems
- Mr. Russ Wertenberg from NASA for his insights into the socio-techno dynamics in Space Programs
- Thanks to Christopher Fry at http://www.clearmethods.com for his implementation of the OPN runtime engine using the Water Programming Language
- Mr. Qiu Yilin for his original concepts in evolvable software systems
- Prof. Paul Carlile for his contribution in using boundary objects as a common currency for socio-technical system interactions
- Dr. Geilson Loureiro, Karen Marais, A-P Hurd for their respective intellectual contribution and technical reviews of this presentation.