Home

A Game Grammar

by Stéphane Bura

sbura@pobox.com

Introduction

This is a reaction to Raph Koster's GDC 2005 presentation: A GRAMMAR OF GAMEPLAY: Game Design Atoms, Can Games Be Diagrammed?
Raph's work has often been a source of inspiration to me, so I hope to give a little bit back with this.

The goal of this document is to present a grammar enabling game designers to describe games in a useful manner. By useful, I mean that this grammar should allow us to:

- Communicate the underlying principles of a game: How are the different parts of the game linked? What kind of interaction is there between the players? Are there winning strategies or exploits? Etc.
- Doodle a game on a napkin, as Raph puts it.

After working on the problem a bit, I came to the conclusion - as Raph posits - that a useful game grammar would mostly describe **constituative rules**, as defined by Katie Salen and Eric Zimmerman in Rules of Play:

> *The constituative rules of a game are the underlying formal structures that exist "below the surface" of the rules presented to players. These formal structures are logical and mathematical. [...]*

01

Ellen Haas

Working at this level also requires some assumptions, such as assuming that all the players follow the **implicit rules** (etiquette). For instance, in a competitive game, all players must play to the best of their abilities to win.

This article mostly talks about how I came up with a diagram for Checkers, since it's the example used in Raph's talk, but you'll find diagrams for other games as well therein.

This grammar could benefit from more research in order to become a generic tool. This is only a first step and I welcome feedback.

Grammar elements

The proposed grammar is heavily influenced by my experience with Petri Nets and cybernetics.
Non-standard Petri Nets (with colored tokens and inhibitors) can describe complex processes and system states with very few elements. It's also very easy to jot down a messy representation of a system just by describing it verbally - using natural language - and associating token and places to resources or states and transitions to "stuff the system does".
A basic understanding of cybernetics also helps with managing the feedback loops that give structure to almost all balanced games.

Because of this heritage, the game diagrams end up looking like flowcharts. However, they are **not** flowcharts of games or rules. They're closer to data-flow representations of processes.

This first iteration of the grammar is composed of the following elements:

- Resource box
- Link
- Transition
- Resource tokens
- Source
- Sink
- Inhibitor link
- "If empty" link
- Variable link
- End state

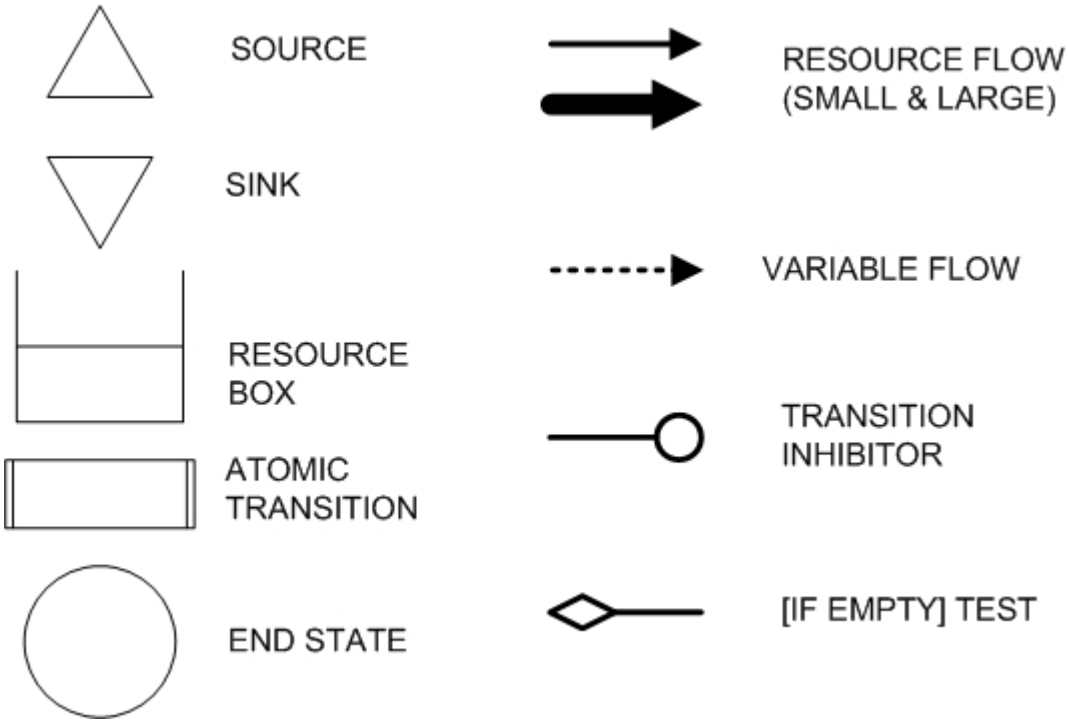


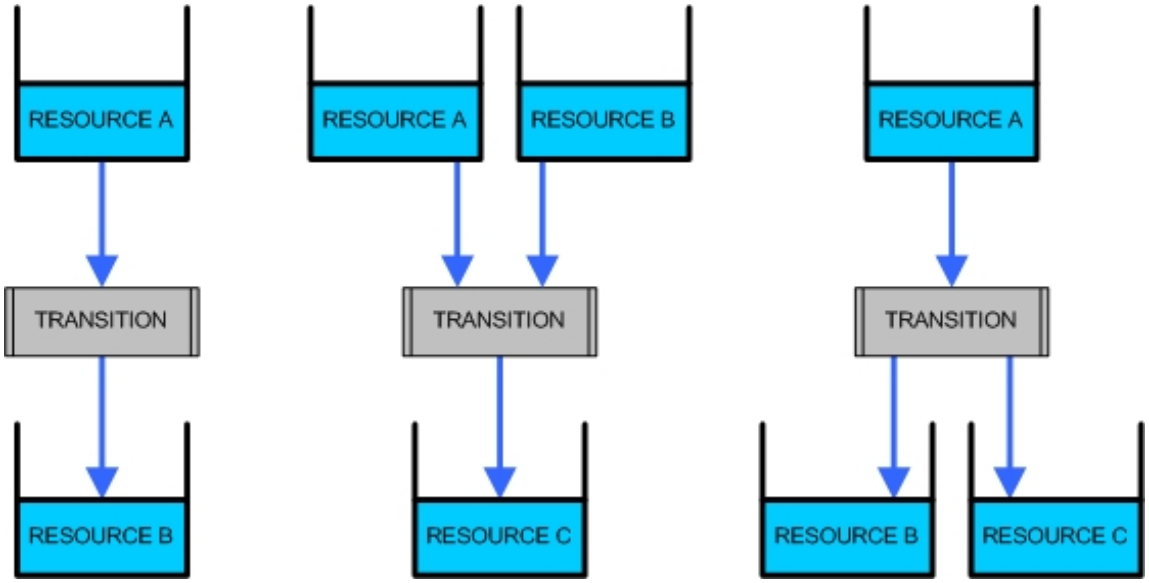
Diagram elements

The four most common elements are resource boxes (hereafter, boxes) and tokens, links and transitions. A box holds tokens. Links are oriented and can only exist from a box to a transition, from a transition to a box, but *not* from a box to a box or from a transition to a transition. A transition **consumes** tokens from all the boxes linked to it and **generates** tokens in all the boxes linked from it. The amount of tokens consumed or generated is specified by each link. In this article, a thin link denotes a small amount of tokens, a thick one, a large amount [we won't need more precise measures for our examples.]. A transition can only be triggered if there are enough tokens to "pay the cost" of each inbound link.

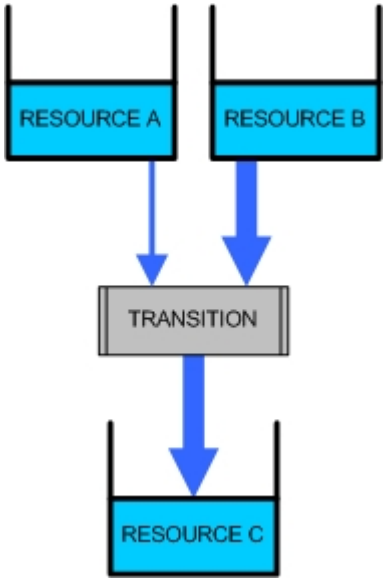
Note: In this article, resource boxes are depicted by actual "boxes" in order to emphasize the downward flow of tokens. Any shape can be used instead. Petri Nets use circles.

It's easier to think in terms of consumption and generation instead of imagining that tokens are somehow transformed from one resource into another, since the latter metaphor breaks down in the

second and third examples below.



Basic transitions



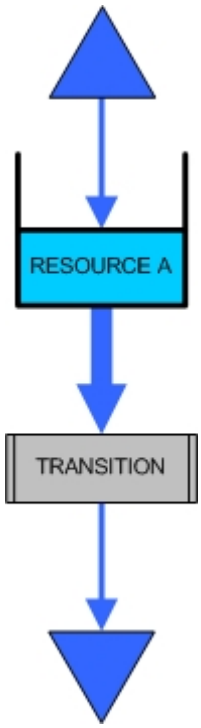
A few "A" tokens and a lot of "B" tokens are needed to trigger the transition in order to generate a lot of "C" tokens

Most of the time, transitions denotes actions in the game. In the following examples of game diagrams, I've tried to find an isomorphism between the atomic transitions and the actions permitted by the **operational rules** (the rules of the game proper), but I didn't let myself be limited by that, as we'll see in the abstraction section below. However, **transitions represent discrete game state changes caused by a player's choice**.

A **source** continuously generates tokens at a rate specified by its outbound link (number of tokens per time unit). Sources are usually linked to boxes but can sometimes be directly linked to

transition, for simplicity, as it's the case in the *Decay of preparation in a skill-based game* ludeme below.

A **sink** consumes tokens. It's just an aesthetic element to avoid having transitions with no outbound links.



Source and sink

An **inhibitor link** links a box to a transition. If the stated amount of tokens is consumed, the transition cannot be triggered during the same time unit. An inhibitors link is denoted by a circle at its end.

An **"If empty" link** originates from a box. It generates its specified amount of tokens each time unit said box is empty of tokens. An "If empty " link is denoted by a diamond at its origin.

The amount of tokens specified by a **variable link** can change each time unit according a function (often a random law) that must be defined or by events outside the magic circle. A variable link is denoted by a dotted line.

An **end state** is an element that can only have inbound links. It usually is linked to a transition but can also be directly linked to a box for simplicity. When one of its inbound links transfers token(s), the game ends. The nature of the end state defines the outcome of the game (victory, stalemate, etc.). An end state is denoted by a circle.

The "If empty" link is mostly used for victory conditions or in situations like the one in diagram A below (if empty then inhibit), which is shorthand for diagram B.

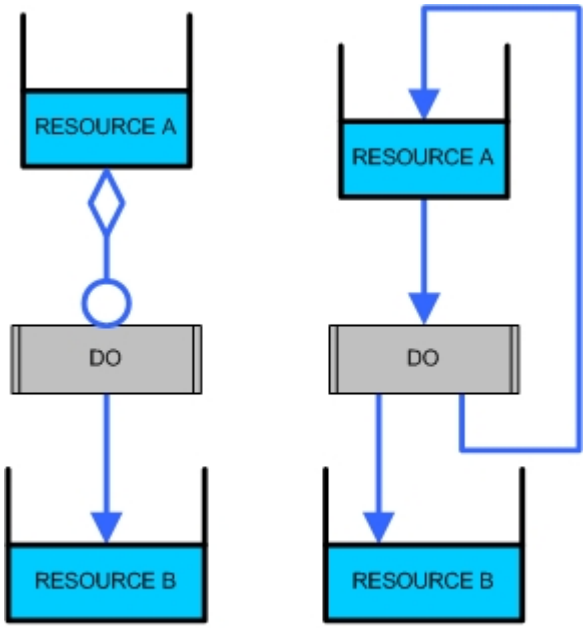
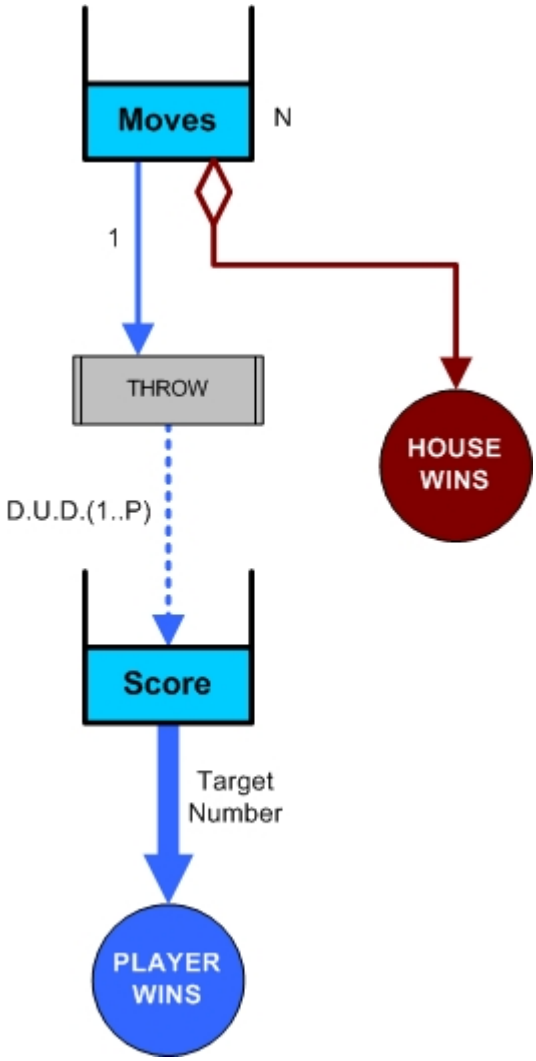


Diagram A Diagram B

Here's an example for a simple game. In this game, the player plays against the house. Each turn, he throws a P-sided die and adds the number he gets to his score, which starts at zero. If his score reaches or exceeds a target number in N turns or less, he wins. Otherwise, the house wins. This game diagram is:



Notes: "D.U.D.(1..P)" stands for *discrete uniform distribution* for a set of integer values from 1 to P, i.e. what a fair die generates.
Colors are used to differentiate the players (blue for player, red for the house).

This game is not very interesting: The lonely transition indicates that there are no choices for the player to make.

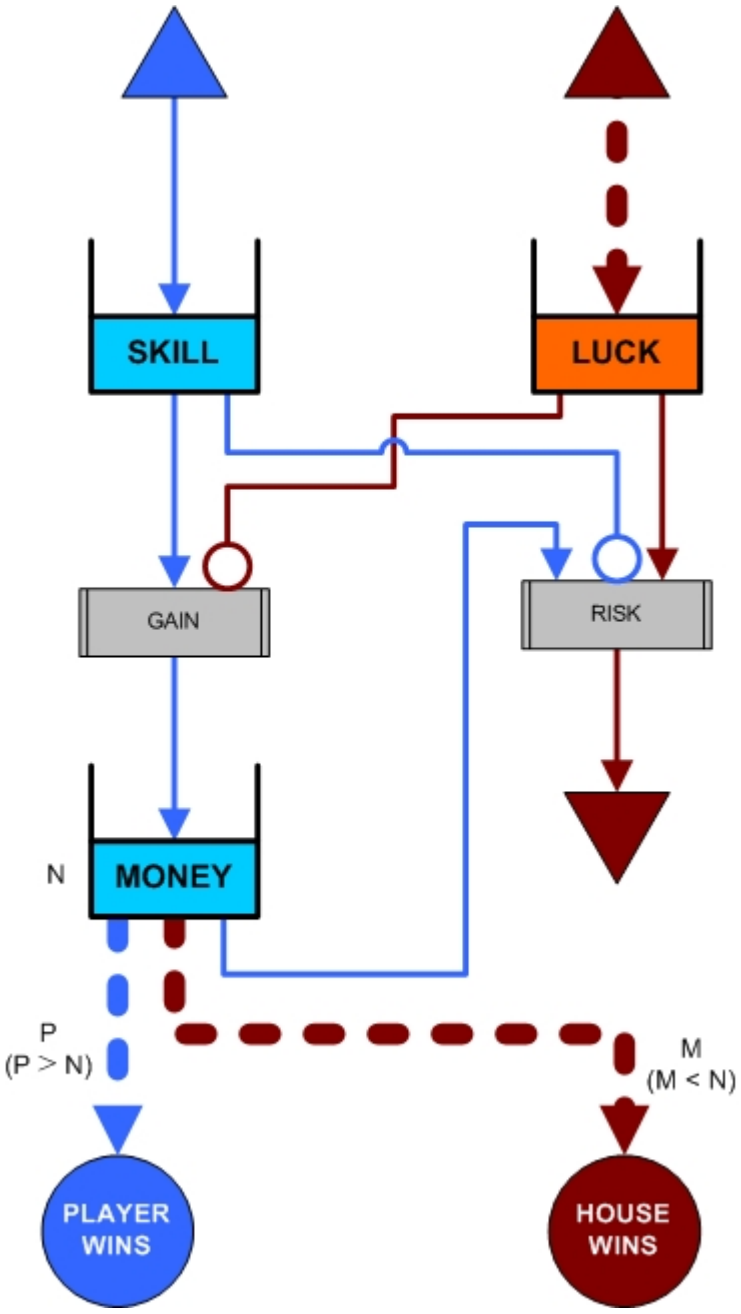
Conceptual Tools

Abstraction

If we look at the example above, we might think that this grammar is well suited for describing operational rules. There even is an isomorphism between the rule elements of the game and its diagram's elements. Actually, it's a mistake to think that way and, I believe, a mental block that has prevented game notations from flourishing up till now. If we were to try to describe Checkers operational rules with such a grammar, we'd end up with an unwieldly diagram, too complex because we'd have to represent within it the mechanisms for movement, capture, promotion, etc. Furthermore, the operational rules don't cover what we really need from such a diagram: the tactical and strategic elements (the concept of sacrifice, preparing multiple captures, showing that Checkers is a game of skill, etc.). So, instead of mapping our diagram elements onto the ones described by the

operational rules, **we'll use abstract elements created specifically for each diagram**. We'll see that some of these elements can be re-used from one game to another, especially the elements related to "Skill", "Luck" and "Preparation" resource tokens.

How far can we go with abstraction? Fairly far. For instance, here's an abstracted view of Blackjack:



Abstract Blackjack

Let's take a moment to understand how this diagram works and what it means. Fisrt off, we see no mention of cards, hitting, standing, 17 or 21. What's going on and how can this diagram explain the constituative rules of blackjack? This diagram is not meant to teach us how to play Blackjack well at the hand level. It deals with how Blackjack works over a longer period of time. It shows that (for the player) Blackjack is a game of resource and risk management.

Blackjack, like all non-competitive casino games, is about one thing only: Can the player beat the odds? So, almost naturally, the relevant resources become Skill and Money (the measure of success) for the player and Luck for the house. The amounts of money a player brings to a game, gains or loses vary widly. Money operates outside the magic circle: the player stops playing when he feels he has won or lost enough. This is represented by the values of the links to the end states (P and M) that the player can dynamically set himself (during the course of the game). If he leaves with a profit, he wins. If not, the house wins. The player "spends" his skill to gain money and avoid risk. The house spends its luck to prevent player gains and to cause him losses.

Time is also abstracted here. There are no turns or hands depicted in the diagram. Resources flow continuously from the sources and are spent simultaneously by both players. Blackjack gives choices to the player because he usually doesn't have enough skill to spend on both transition all the time. He must gamble that when he doesn't inhibit the risk transition, the house's luck has run out. We see that, apart from the sources and the resources names, the diagram is symmetrical regarding both opponents (I could have made that clearer by using a box for the house's money instead of a sink). So, if the house wants to win more often than the player, it just has to make sure that, over time, its source produces more tokens than the player's. Casinos achieve this by carefully choosing the random laws for their sources and by capping the flow of players' skill sources (for instance, by forbidding card counting).

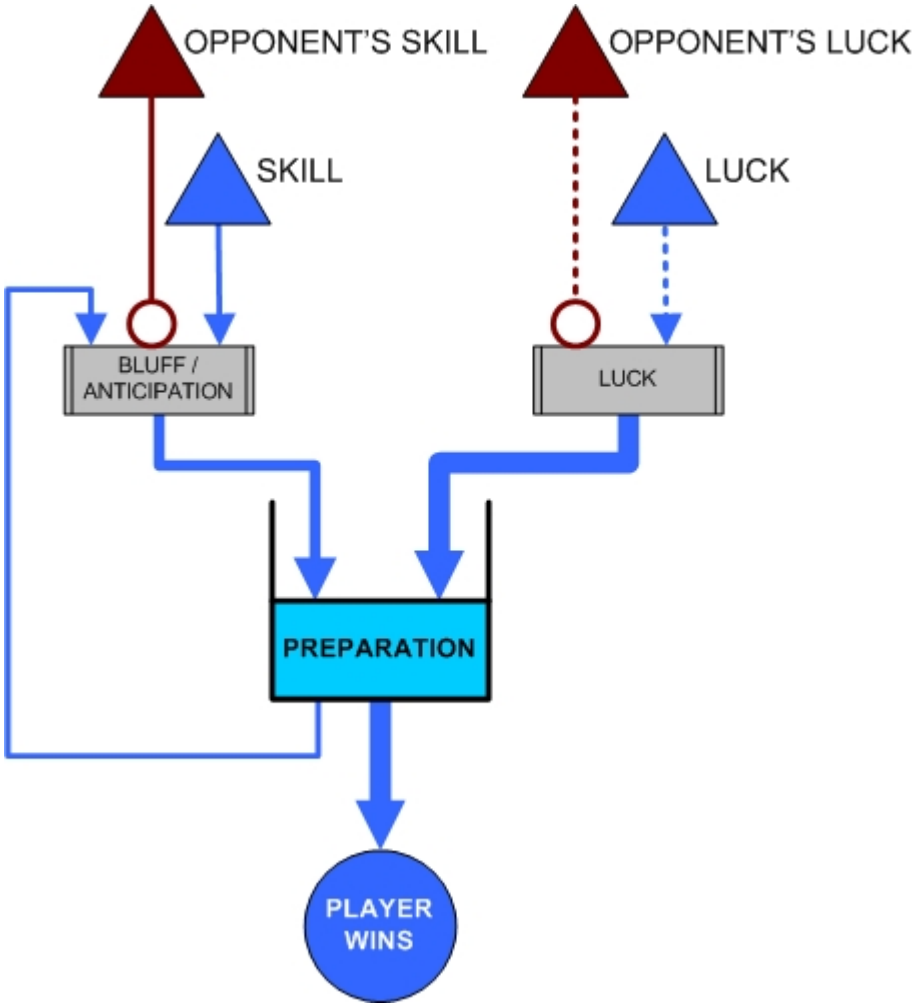
As said above, this diagram is not very helpful in describing how the actual game plays. If needed, the actual hand playing diagram can be described elsewhere, as a sub-game to the main game (beating the odds). The corresponding sub-diagram would have sources and sinks that represent tokens spent and generated in the main diagram. Such a sub-diagram isn't the detailed description of a transition. Here, for instance, it would describe both transitions at once.

Note: Depending on the actual link values and random law, the above diagram can be used for most non-competitive casino games.

Diagram folding in symmetrical games

Colored Petri Nets can represent very complex symmetrical processes (such as the exchanges between two computers, alternatively client and server) with only one instance of the symmetrical part, using a technique called "folding". I'll use a similar idea here for diagrams of games assigning symmetrical roles to each player (Checkers, Chess, Parcheesi, etc.) or symmetrical roles at the constituative level: I'll represent the game as seen from the point of view of one of the players. The representation of other players' elements is only necessary when resources are transferred from one to another. Thus, such a game diagram is only part of the complete picture: One needs as many diagrams as there are players to describe a game state. But you need only one to fully describe the process.

Here's an example for the "Rock - Paper - Scissors" game:

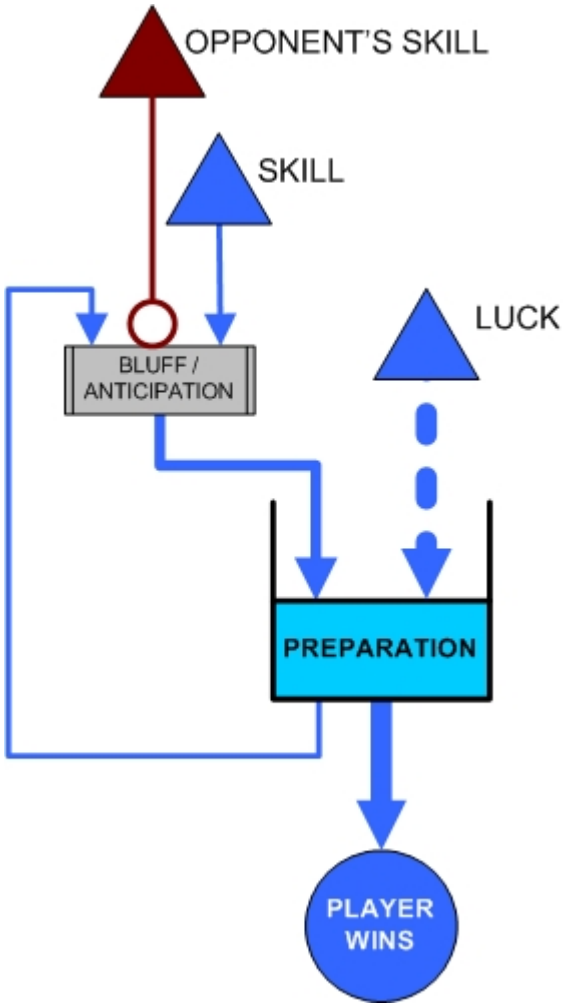


Folded RPS diagram

Again, there's no mention of turns or of circular relative order of symbols. These rules have little to do with how the game is played: it's a bluffing game (thin Skill links) with a strong random component (thick Luck link). If the player is more skilled than his opponent, his "bluff" transition will sometimes trigger (if the Skill tokens flow of his opponent is not high enough to always inhibit the transition).
There is only one end state. The first player to reach his, in his own diagram, wins.

Incidentally, the RPS diagram hints at the best strategy: Be lucky and play randomly, as bluffing is not very cost effective. Another option is to use the metagame to begin the game with some tokens in the Preparation box ("Rock is strong!").

This folded diagram contains the skill sources of both players. Thus, the player represented here also has his skill source in his opponent's diagram, linked with an inhibitor to his opponent's bluff transition. This means that there is a choice where to allocate the skill tokens. In RPS, it represents how seemingly predictable a player tries to be before springing a trap.
The opponent's luck is included to signify that both players can't be lucky at the same time. If we admit this as a general rule, we can dispense from including the opponents' luck hereafter. We can then replace the upper right part of the diagram with a thick variable link from the player's luck source:



Revised "Rock Paper Scissors"

Preparation

The Preparation box in the RPS diagram does not reflect the number of games won by the player. It's an abstract resource. Preparation is one of the "ingredients" of Raph's grammar:

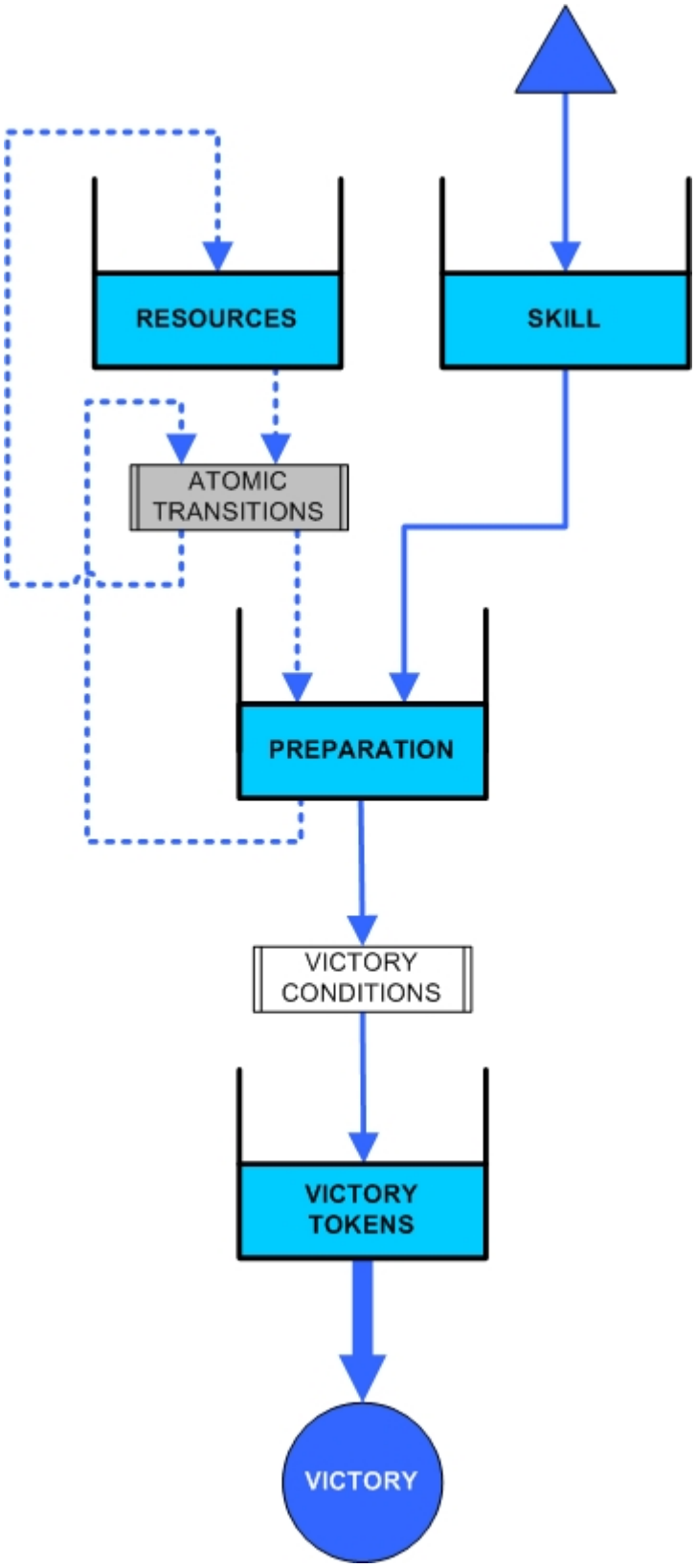
- Territory
- Preparation
- Core mechanic
- Range of challenges
- Choice of abilities
- Skill required
- Variable feedback
- Dealing with the Mastery Problem
- Cost of failure

Preparation stands for logical connections between atomic transitions and prior choices made that influence the next atomic transition.

In a simplified version of a diagram, all the preparation tokens go into the same box where they are differentiated by color. A more comprehensive diagram should have one preparation box for each color. This is helpful when gameplay relies on a lot of small preparations of different natures: What

is lost and gained when a played jumps in a FPS? What is the difference between a standard block and a low block in a combat game? Etc.
Using colors for the preparation tokens and the transitions that generate and consume them let us describe high-level processes: Chaining (plot sequence, combos), recursion (multi-level resources combination in crafting), or parallelism (attack & defense, skill trees and stats).
Yet, for the purpose of producing napkin diagrams, treating preparation as one resource is often enough.

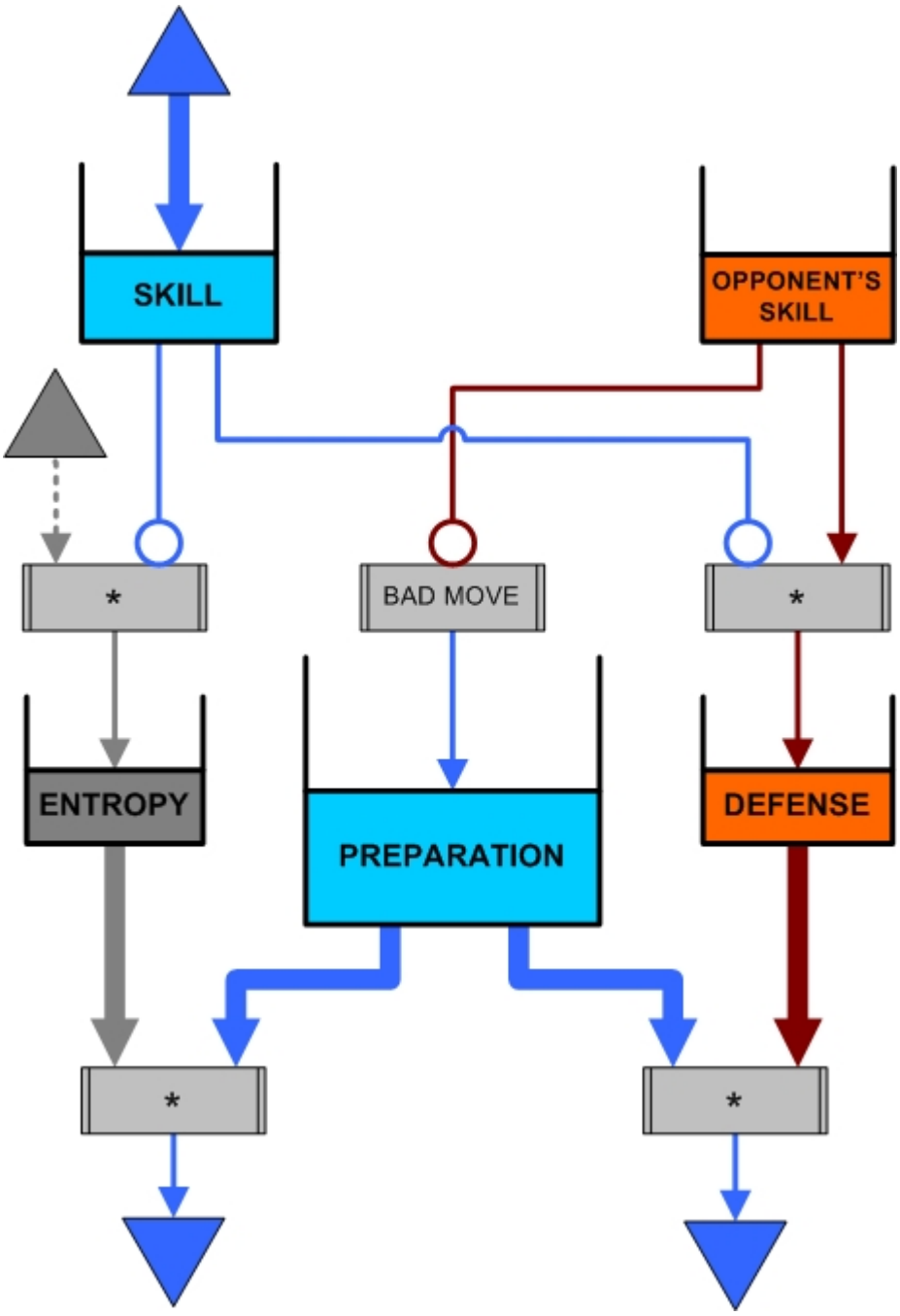
In a purely skill-based game, the player with the highest skill must win more often than his opponent(s) - if no alliances between weaker opponents are possible. How do we convey that? The players accrue preparation tokens using skill and atomic transitions. Those tokens are needed to trigger a victory condition transition that leads to a Victory end state:



Simplified diagram for a skill-based game

Ludeme: Decay of preparation in a skill-based game
However, victory of the most skilled player is not automatic. He can make bad choices (which give preparation tokens to his opponent), focus too much on defense or fail to act at the right time. The

sub-diagram below reflect this. It is part of all diagrams for skill-based games (even non-symmetrical ones).

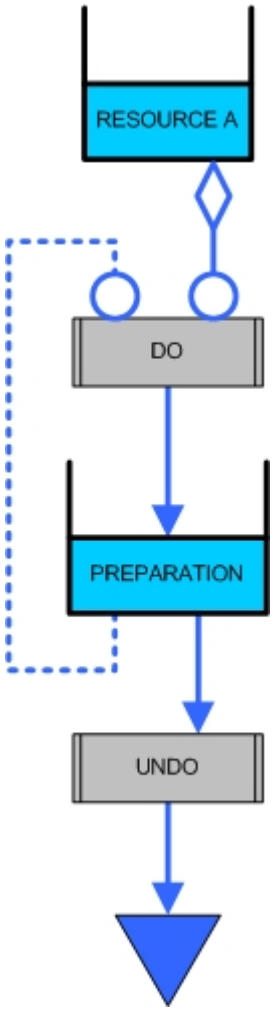


Ludeme: Decay of preparation in a skill-based game

Note: this ludeme can easily be adapted for more than two players.

Risk-Free Transitions

A risk-free transition earns preparation tokens to a player without costing him anything beyond the time it takes to execute it. Risk-free transitions are often reversible. They look like this in a diagram:



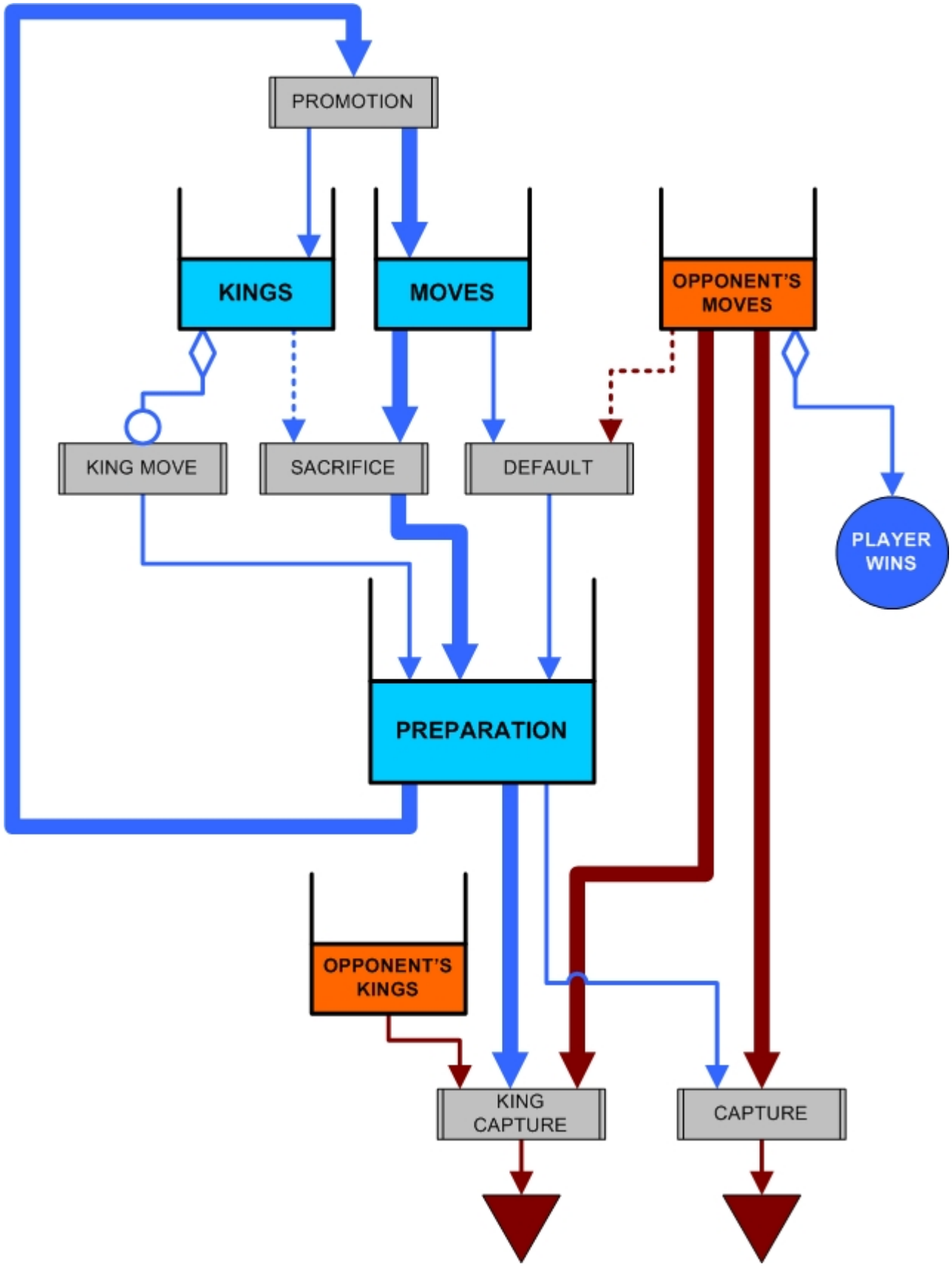
Risk-free transition

Note: The dotted line-inhibitor link clumsily represents that, sometimes, special conditions are required for a risk-free transition to become available. Another way to show that could be to expand the grammar with conditional transition or, probably better, add a layer to the diagram for the operational rules. The transitions in that layer could link from and to all the boxes in the diagram, would have priority over those in the constituative rules layer, and trigger automatically when their inputs are valid.

For instance, outside of combat, avatar movement in an MMOG is a risk-free transition.

Checkers

Here the folded diagram for Checkers:



Checkers - the napkin version

A player wins when his opponent can no longer make a legal move. The relevant resource is then Moves, the only box containing tokens at the beginning of the game. The player uses Moves tokens

to generate preparation tokens. This reflects the rule that normal pieces can only move forward, thus decreasing the number of potential moves each time they do. The dotted line from the opponent's Moves box means that some default move sometimes block the opponent's pieces. However, the most efficient way to make him lose Moves tokens is through capture, that is made easier through sacrifice, which itself costs a lot of Moves tokens.

A king move in checkers is very close to a risk-free transition. Since it produces preparation tokens and can lead to victory, playing with one's kings would clearly constitute an exploit if there weren't rules to counter-balance it. The first one of these rules is that kings can be captured. The second one addresses the fact that since a king move doesn't cost Moves tokens, this could lead to infinite games. Thus, the rules of Checkers specify that a given board configuration cannot be repeated more than a limited number of times (this operational rule is not represented in the diagram).

Conclusion

This is only an early stab at the game diagramming problem.
If this model seems to be adequate for communicating about simple games, more complex ones would prove too problematic at this time. Furthermore, since it does not take into account the operational rules of the game, one cannot reverse-engineer changes to a game diagram to find corresponding changes in the game rules.

To improve the grammar, we'd need rules about how to break down a game into sub-diagrams, a library of ludemes with agreed upon inputs and outputs, some knowledge about what the relevant resources are for different categories of games, a way to represent overall difficulty, techniques to link operational and constituative rules, etc.

Baring that, this grammar will remain a simple descriptive tool instead of an analytical or even a design tool.
So, if you have ideas about improvements, let me know.

March 2006

Home

A Game Grammar

Bura, S.

01 Ellen Haas Page 1

13/11/2018 18:57