Chapter 2

Linear Logic for Narrative Structure

2.1 Introduction

What makes up a story? In computer science research, this question has been invoked repeatedly by artificial intelligence and computational linguistics researchers. The motivation for answering it includes several potential applications: autonomous agents that can collaborate on building stories with people (*interactive storytelling*) [CCM02, MS99]; computational understanding of narrative structure from natural language text (*narrative extraction*) [CJ08]; and the automatic generation of stories, a kind of investigation on the boundaries of computational creativity (*narrative generation*) [Mee76].

Besides these computational applications, understanding narratives can lend insight into human learning and cognition: narratives are an important way that humans make sense of the world around us [Her03, Bru90]. Narrative researchers have achieved consensus that *causality* is an important aspect of stories for narrative models to capture [TS85, Ada89, MW00, Dah12], even if it only barely scratches the surface of interesting facets of narrative [RFW09]. To understand causality in stories, we first need to be able to break them apart into events between which causal relationship may be found.

As described in Chapter 1, we are interested not only in the structure of existing stories but in systems that allow or describe *collaborative construction* of story. Understanding a story as made of separable pieces is a good first step for this purpose, too, since narrative play involves at least one human mediating between story components.

In this chapter, we aim to give a foundational account of narrative systems. We are interested in representations of narrative that can be interpreted by a computer for the sake of analysis, generation, and interaction, while also using high-level language constructs that are not bound by particular machine models.

We present *intuitionistic linear logic* (ILL) as a formalism to serve as the foundation for modeling narratives and their structure. Linear logic, first devised by Girard [Gir87], was originally suggested as a suitable representation for narratives by Bosser et al. [BCC10], whose work we largely recapitulate here.

Figure 2.1: Fabula versus Sujet in Memento

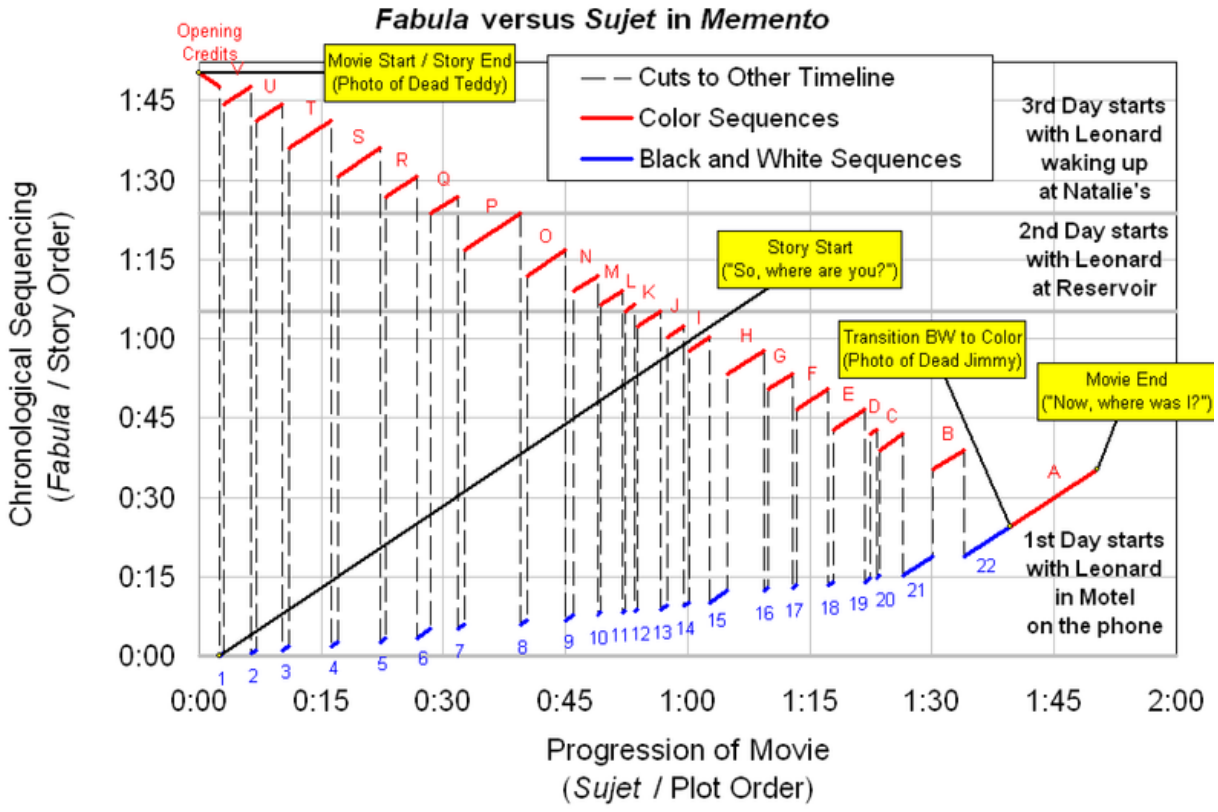


Diagram created by Steve Aprahamian, from `https://en.wikipedia.org/wiki/File:Memento_Timeline.png`, used on the conditions of the Creative Commons license described therein.

2.2 Modeling Narratives

We aim to give a composable formalism for narrative, so we must make clear what aspects of narrative we aim to make formal. In particular, we make the distinction between *fabula* and *sujet* made by Russian formalists in the 1920s [Bro74]: the fabula is the plot events, the "what happens" part of a story, whereas the *sujet* is the *telling* or orchestration of story elements into a (usually linear) digestible tale. In this work we aim only to formalize the *fabula*, while recognizing that there are many interesting research questions in the formalization of sujet and relating fabula to sujet (see, for example, the conflicting structure between the two in the film *Memento*) [Nol00] (see Figure 2.1).

The formal modeling of fabula has a long history in AI research, but until recently has depended on ad-hoc, genre-specific ontologies such as Propp's "morphology of a folktale" [Pro10]. Instead, we would like the particular genre or conflict structure to be *reflected* in the formalism, but not *constrained* by it. The core structure we would like to constrain with the formalism instead includes things like causality, action, and change. These properties are essential to enforcing *coherence* of a narrative [You99].

Folktales as a domain contain many interesting examples for the study of fabula,

since their tellings vary with every storyteller, but their component events remain relatively consistent. Let us consider as an example the folktale *The Three Little Pigs*. This folktale consists of the following narrative events:

1. The mother pig sends her three baby pigs away from home, warning them about a big bad wolf.
2. The first pig builds a house out of straw.
3. The second pig builds a house out of sticks.
4. The third pig builds a house out of bricks.
5. The big bad wolf visits the straw house and blows the house down.
6. The big bad wolf visits the stick house and blows the house down.
7. The big bad wolf visits the brick house and attempts to blow it down, but cannot.

Even within formalization of fabula, there are many conflicting ideas about what parts of this story to include in narrative state and how to model relationships between these narrative elements. Some candidates for narrative components include:

- Characters in the story: the three little pigs, their mother, and the big bad wolf.
- Resources or props in the story that are needed to drive the story forward, such as the straw, sticks, and bricks that the pigs use to build their houses, and subsequently the houses themselves.
- Relationships between any of the above entities: feelings of family and allyship between the pigs; feelings of enmity between the pigs and the wolf; locations and possessions of characters.
- Scenes, or story events, which require availability of the above story elements and may change them. *The first pig builds a house out of straw* might be understood as changing the state of the pig, the straw, and the house.
- Establishment of initial configurations for characters, resources, and relationships. The first line of the story establishes the three little pigs, their mother, and the wolf as characters in the story.
- Endings, or narrative goals established by the author. "Happily ever after" is a common goal ending for fairy tales; in variants of The Three Little Pigs, the defeat of the wolf is an important constant.

Finally, we may also need to represent relationships between *scenes*. These relationships may not be represented explicitly by an author, but they are important for computational applications of story. For instance, in the context of interactive stories, we may need to include or derive links between story scenes to represent alternatives. In the context of story *analysis*, we may care about the ability to determine *causal* relationships between scenes, as mentioned in the introduction.

2.2.1 Linear Logic by Example

Modeling a story in logic means mapping the above concepts to the constructs of logic, which is to say *propositions* and the notion of *derivability* between propositions. Propositions can either be *atomic* or *compound*. An atomic proposition is something like "The straw house is standing," which we might abbreviate with the notation $\mathsf{straw\_house}$. Compound propositions are how we combine atomic propositions into more complex statements, analogous to English usage of *and*, *or*, and *implies* to combine statements. The logical analog of these conjoning words are the *connectives*.

The following linear logic proposition could model the story event of the wolf blowing down the straw house:

$$\mathsf{wolf} \otimes \mathsf{straw\_house} \multimap \mathsf{wolf}$$

The atomic propositions shown here are wolf and $\mathsf{straw\_house}$. The connectives are $\multimap$ ("lolli," linear logic implication) and $\otimes$ ("tensor," linear logic conjunction). $\otimes$ binds more tightly than $\multimap$, so the rule can be read as, "If the wolf is present and the straw house is standing, then transition to a state where (only) the wolf is present."

Note that this reading avoids the word *implies* or an *if-then* reading, although we asserted that $\multimap$ is linear logic's form of implication. In linear logic, derivability is defined in such a way that it represents *state change* more than inference about permanent truth. This mechanism lets us treat propositions as *resources*, or in this case, slices of narrative state, and implication (embodied by derivability) can be read as replacement of one piece of state with another.

Logical connectives need not always form compound propositions: for instance, the "connective" 1 represents the empty or null resource, and we can use it to model the removal, rather than replacement, of a piece of state. For instance, the following proposition models removing the wolf from the story:

$$\mathsf{wolf} \multimap 1$$

On the other hand, this proposition models introducing the wolf into the story from out of thin air:

$$1 \multimap \mathsf{wolf}$$

Another connective in linear logic, $\&$ ("with"), allows us to represent choices between alternatives. The following proposition models a transition from the pig state (representing a pig with no house) to either a straw, stick, or brick house:

$$\mathsf{pig} \multimap \mathsf{straw\_house} \,\&\, \mathsf{stick\_house} \,\&\, \mathsf{brick\_house}$$

Note that we could equally well represent this choice-yielding event as instead a choice between events:

$$(\mathsf{pig} \multimap \mathsf{straw\_house}) \,\&\, (\mathsf{pig} \multimap \mathsf{stick\_house}) \,\&\, (\mathsf{pig} \multimap \mathsf{brick\_house})$$

So far, all of these propositions are just syntax; we have not attempted to explain what they mean in terms of logical derivability, except at an intuitive level. But one of the more important omissions we have made is the status of these propositions that represent story events: how do we intend to compose them?

Linear logical implications do not really accurately model concrete story events: they instead describe a *possible* event that could occur given the fulfillment of the antecedent. We are instead describing a story *world*, a set of possibilities that we intend to give rise to a story or set of stories through logical deduction.

In this sense, we can imagine composing implications $E_1 \dots E_n$ representing story events as the compound proposition

$$E_1 \otimes \dots \otimes E_n$$

meaning that all events are simultaneously available for use. However, if we have two events $A \multimap B$ and $A \multimap C$, and A is some narrative state that we only expect to happen once, then one of the events will go unused, which is disallowed in linear logic.

Sometimes we *want* to enforce the usage of a given event because we are interested in *narrative drive*: the idea that an author needs certain scenes to occur to fulfill certain dramatic intent. In this case, the tensoring-together of narrative events should suffice, and any conflicting scenes would need to be carefully crafted as alternatives (e.g. $A \multimap B \,\&\, C$).

On the other hand, sometimes we are interested in modeling a more exploratory possibility space where some events may not take place at all, and others may take place multiple times. The unary connective ! gives us exactly this meaning: $!A$ is a proposition A that may be ignored or repeated. Thus, we can represent the more exploratory story world as follows:

$$!E_1 \otimes \dots \otimes !E_n$$

We hereafter refer to a proposition of the form $!(A \multimap B)$ as a *rule*. Since we may refer to rules multiple times in a story, we will often name them using the syntax $r : R$, where r is a name and R is a rule.

In the next section, we seek to clarify the meaning of these propositions and simplify our representations by explaining them in terms of a few carefully-considered, meta-logical principles.

2.3 Linear Logic

Any representation of fabula needs to include models of *action* and *change*, phenomena for which many formalisms have been investigated over several decades [MH69, KS89, Lam94]. The cited works effectively solve the variability problem by indexing every proposition with a time parameter and modeling actions as incrementing that parameter. As a consequence, they all need to explicitly manage *inertia*, or the fact that when a

rule describes changes of one part of the world state from time t to $t+1$, the remaining parts of the world state are updated in $t+1$ to retain their state at time t.

Linear logic is an approach to the logical modeling of action and change that does not depend on explicit time indexing. This approach has the advantage that it is possible to conceive of distinct parts of the state changing independently of one another.

We note that, unlike Girard's original formulation of logic, our chosen formulation is *intuitionistic* in that we only consider sequents $\Delta \vdash A$ where A is a single proposition, rather than a disjunction of multiple propositions. The intuitionistic version of linear logic (ILL) was first presented by Girard and Lafont [GL87]. We select it mainly for its computational interpretations as described by Andreoli [And92], although we note that this choice may not be canonical, and classical linear logic bears investigating as an alternative in future work.

Intuitionistic linear logic was later reformulated [CCP03] according to the methodology of logic design espoused by Gentzen and Martin-Löf [Gen35, ML96] in which inference rules operate over *judgments*, or meta-logical syntax pertaining to propositions, and inference rules also obey certain meta-theoretic properties that make soundness of the logic easy to establish, and proof search within the logic easy to automate. This methodology, which includes giving a *sequent calculus* formulation of the logic, is described below.

First, we assert that to be a *logic* means to have a notion of *consequence*:

$$A_1, \ldots, A_n \vdash A$$

The above syntax can be read, "A is a consequence of $A_1, \ldots, A_n$," where A and all A_i are propositions (defined by the particular logic). Such a statement is called a *sequent*, and an arbitrary collection of assumptions $A_1, \ldots, A_n$ is often notated as Δ and called the *context*.

A logic may have *inference rules* of the form

$$\frac{J_1 \quad \ldots \quad J_n}{J}$$

where each J and J_i is a sequent, the J at the root of the rule is the *conclusion* sequent, and each J_i is a *premise* that must be satisfied in order for the inference to be valid.

A *proof* of sequent J is a composition of inference rules forming a tree whose leaves are rules with no premises and whose root is J.

Finally, to count as a logic, the notion of consequence must be *transitive*; that is, if it is possible to build proofs of $\Delta \vdash A$ and $\Delta', A \vdash C$, then we can also build a proof of $\Delta', \Delta \vdash C$.

This property, having a transitive notion of consequence, is also known as Admissibility of Cut [Gen35]. It is a meta-property of a system of inference rules and part of what gives it meaning. There are other such properties, called structural properties, that have often been previously considered part of what it means to be a logic:[1]

[1]In the literature, the terminology "consequence relation" has sometimes implied these properties. We depart from this tradition by using "consequence" without including weakening and contraction by

Weakening. If $\Delta \vdash C$ then $\Delta, A \vdash C$: in other words, adding unneeded assumptions does not affect derivability.

Contraction. If $\Delta, A, A \vdash C$ then $\Delta, A \vdash C$: in other words, having two copies of a proposition available is not any better than a single copy.

These properties encapsulate the idea that in a proof from assumptions, each assumption may be used arbitrarily many times to reach the goal. Linear logic, however, denies these properties, leading to a "use exactly once" treatment of assumptions in a proof. Put another way, if ordinary logic treats the context Δ as a *set* of assumptions, in linear logic Δ can be thought of as a *multiset* [Gir95] i.e. the multiplicity of assumptions matters.

Thus linear logic's notion of consequence $\Delta \vdash A$ embodies a sort of conservation property, and can be read "Resources Δ may be transformed into A." Connecting this notion back to narrative, we can think of Δ as an initial narrative situation and A as a narrative ending.

To include the ! connective in linear logic, we need to reintroduce a context that *is* subject to weakening and contraction. If we change our basic judgment (sequent form) to

$$\Gamma; \Delta \vdash A$$

where assumptions in Γ may be weakened and contracted, then we have the machinery we need to define !, effectively reintroducing ordinary, persistent notions of logical fact into the by-default resource-oriented logic. In this sense, intuitionistic linear logic can be thought of as a refinement on ordinary intuitionistic logic.

In this more foundational account of the logic, we can rewrite our story world representation from Section 2.2.1 as simply a specification for the context Γ: instead of imagining

$$!E_1 \otimes \ldots \otimes !E_n$$

as an assumed proposition in Δ, we can instead stipulate that

$$\Gamma = E_1, \ldots, E_n$$

We can now describe the mapping between story world and logical sequent that we will use as our primary model of representation going forward. The sequent $\Gamma; \Delta \vdash A$ represents a story world with rules (and persistent facts) in Γ, initial narrative configuration described by Δ, and narrative goal (or story ending) A.

Now that we have sketched the judgmental framework in which we intend to model story worlds, we can finally give the formal definition of linear logic as a sequent calculus.

default.

2.3.1 Intuitionistic Linear Logic: Sequent Calculus

In a sequent calculus, every connective is defined by two rules: instructions for how to prove it when it appears on the right-hand side of the turnstile ($\vdash$) in a sequent, and instructions for how to *use* it when it appears on the left-hand side of the turnstile. These instructions come in the form of *left* and *right* inference rules. We now present the inference rules of linear logic that define the connectives we need for narrative modeling.

The $\otimes$ connective models simultaneous conjunction between two resources:

$$\frac{\Gamma; \Delta_1 \vdash A \quad \Gamma; \Delta_2 \vdash B}{\Gamma; \Delta_1, \Delta_2 \vdash A \otimes B} \otimes R \qquad \frac{\Gamma; \Delta, A, B \vdash \gamma}{\Gamma; \Delta, A \otimes B \vdash \gamma} \otimes L$$

From a proof search perspective, we read these inference rules from the bottom up. So the right rule says that a proof of $A \otimes B$ can be formed from a context that can be partitioned into pieces Δ_1 and Δ_2, which can prove A and B respectively. The left rule says that an assumption of $A \otimes B$ can be interpreted as two separate assumptions, A and B.

Note that the persistent context Γ is simply repeated in every sequent in this rule (i.e. from a proof search perspective, it is passed unchanged to both subgoal sequents). The Γ context will be "carried through" in this manner for all inference rules that do not refer to it; that is, all rules except those defining !.

The rules for the tensorial unit, 1, are as follows:

$$\frac{}{\Gamma; \cdot \vdash 1} \, 1R \qquad \frac{\Gamma; \Delta \vdash \gamma}{\Gamma; \Delta, 1 \vdash \gamma} \, 1L$$

The null resource may be proven only when the linear context Δ is empty, and it may be used by simply removing it from the context.

The rules for implication ($\multimap$) follow a similar pattern to those for $\otimes$, except inverted with respect the the left and right rule:

$$\frac{\Gamma; \Delta, A \vdash B}{\Gamma; \Delta \vdash A \multimap B} \multimap R \qquad \frac{\Gamma; \Delta_1 \vdash A \quad \Gamma; \Delta_2, B \vdash \gamma}{\Gamma; \Delta_1, \Delta_2, A \multimap B \vdash \gamma} \multimap L$$

The right rule for $\multimap$ says that to prove $A \multimap B$, add A to our set of assumptions and work on proving B. The left rule requires again that we *partition* the context into the part that proves the antecedent A, and another part that, when given the consequent B, continues to prove the original goal.

The inference rules for $\&$ are given below – but note that we do not use this connective in the remaining chapters of the thesis, so understanding it is optional.

$$\frac{\Gamma; \Delta \vdash A \quad \Gamma; \Delta \vdash B}{\Gamma; \Delta \vdash A \& B} \, \& R \qquad \frac{\Gamma; \Delta, A \vdash \gamma}{\Gamma; \Delta, A \& B \vdash \gamma} \, \& L_1 \qquad \frac{\Gamma; \Delta, B \vdash \gamma}{\Gamma; \Delta, A \& B \vdash \gamma} \, \& L_2$$

The right rule says that to prove $A \& B$, we need to be able to prove A and B *from the same context* Δ. The left rule says that if we have a choice $A \& B$, we can make either selection to continue the proof.

The rules for ! are the ones that interact with the persistent context Γ:

$$\frac{\Gamma; \cdot \vdash A}{\Gamma; \cdot \vdash !A}\ !R \qquad \frac{\Gamma, A; \Delta \vdash \gamma}{\Gamma; \Delta, !A \vdash \gamma}\ !L$$

The $!R$ rule says that a resource A can be thought of as a fact $!A$ so long as it is proven *only* with factual propositions (the ones in Γ). Thus the rule enforces that the linear context Δ must be empty.

The left rule allows us to move a $!A$ from the linear context into the persistent context as just A.

Finally, there are two structural rules that interact with the contexts Δ and Γ. One of these is the copy rule:

$$\frac{\Gamma, A; \Delta, A \vdash \gamma}{\Gamma, A; \Delta \vdash \gamma}\ \mathsf{copy}$$

This rule allows us to copy any persistent assumption A into Δ (without also removing it from Γ, meaning we may do this as many times as needed).

Finally, we need a structural rule for proving (and equivalently using) atomic propositions:

$$\frac{}{\Gamma; p \vdash p}\ \mathsf{init}$$

This rule allows us to finish a branch of proof search by observing that the context and goal match up exactly (modulo persistent assumptions Γ).

We could imagine a more general init rule

$$\frac{}{\Gamma; A \vdash A}\ \mathsf{init}'$$

where A can be any proposition, not just atomic p, but it turns out that this rule is *admissible* in a well-designed logic, i.e. the rules already allow it for any given A. Furthermore, testing for its admissibility is a way to determine that the logic is well-designed; it is considered an internal completeness property.

2.3.2 Some Derivation Examples

We can see each of the connectives $\multimap$, $\otimes$, $\&$, and ! as internalizing some property of the judgmental framework: $\multimap$ internalizes consequence, $\otimes$ internalizes the comma for conjoining contexts, $\&$ internalizes a choice between two alternatives, and ! internalizes persistence.

In future chapters of this thesis, we will use all of these connectives frequently except for $\&$; in fact, refinements on this formalism will not include $\&$. We omit $\&$ because in settings where all implications are rules (live in the persistent context), occurrences of $\&$ to the right of a rule can be accounted for by adding multiple rules. In other words,

a rule of the form $A \multimap B \,\&\, C$ can be replaced by two rules, $A \multimap B$ and $A \multimap C$, in a sound and complete way. [2]

We now prove this fact—that $\&$ is an unneeded connective in the context of persistent rules—which will also serve as an example of how to put the inference rules above together into proofs.

Claim:

$$!(A \multimap B \,\&\, C)$$

is interderivable with

$$!(A \multimap B) \otimes !(A \multimap C)$$

That is, we need to show that

$$!(A \multimap B \,\&\, C) \vdash !(A \multimap B) \otimes !(A \multimap C)$$

is derivable, as is its converse

$$!(A \multimap B) \otimes !(A \multimap C) \vdash !(A \multimap B \& C)$$

Recall that a *proof* is a composition of inference rules forming a tree whose leaves are rules with no subgoals and whose root is the sequent being proved. We form a proof by applying inference rules to the current goal sequent, resulting in a new set of goal sequents (the premises to the rule), until there are no more premises.

The first derivation works as follows:

Let $\Gamma = A \multimap B \,\&\, C$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma; A \vdash A} \quad \dfrac{\dfrac{}{\Gamma; B \vdash B}}{\Gamma; B \& C \vdash B}\,\&L_1}{\Gamma; A \multimap B \& C, A \vdash B}\multimap L}{\Gamma; A \vdash B}\,\mathsf{copy}}{\Gamma; \cdot \vdash A \multimap B}\multimap R}{\Gamma; \cdot \vdash !(A \multimap B)}\,!R \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma; A \vdash A} \quad \dfrac{\dfrac{}{\Gamma; C \vdash C}}{\Gamma; B \& C \vdash C}\,\&L_2}{\Gamma; A \multimap B \& C, A \vdash C}\multimap L}{\Gamma; A \vdash C}\,\mathsf{copy}}{\Gamma; \cdot \vdash A \multimap C}\multimap R}{\Gamma; \cdot \vdash !(A \multimap C)}\,!R}{A \multimap B \,\&\, C; \cdot \vdash !(A \multimap B) \otimes !(A \multimap C)}\otimes R}{\cdot; !(A \multimap B \,\&\, C) \vdash !(A \multimap B) \otimes !(A \multimap C)}\,!L$$

The leaves of this derivation are all of the form $\Gamma; A \vdash A$ which we know to be derivable for any proposition A by way of the identity admissibility theorem.

The second derivation works as follows:

Let $\Gamma = A \multimap B, A \multimap C$ in the derivation below.

[2]Rules of the form $A \& B \multimap C$, on the other hand, do not have a direct translation into the $\&$-less fragment. However, we have not encountered a need for such rules in any of the examples we investigate.

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{\Gamma; A \vdash A} \quad \overline{\Gamma; B \vdash B}}{\Gamma; A \multimap B, A \vdash B} \multimap L}{\Gamma; A \vdash B} \text{copy} \qquad \dfrac{\dfrac{\overline{\Gamma; A \vdash A} \quad \overline{\Gamma; C \vdash C}}{\Gamma; A \multimap C, A \vdash C} \multimap L}{\Gamma; A \vdash C} \text{copy}}{\Gamma; A \vdash B \,\&\, C} \,\&R}{\Gamma; \cdot \vdash A \multimap B \,\&\, C} \multimap R}{A \multimap B, A \multimap C; \cdot \vdash !(A \multimap B \,\&\, C)} \,!R}{\cdot; !(A \multimap B), !(A \multimap C) \vdash !(A \multimap B \,\&\, C)} \,!L^2}{\cdot; !(A \multimap B) \otimes !(A \multimap C) \vdash !(A \multimap B \,\&\, C)} \otimes L$$

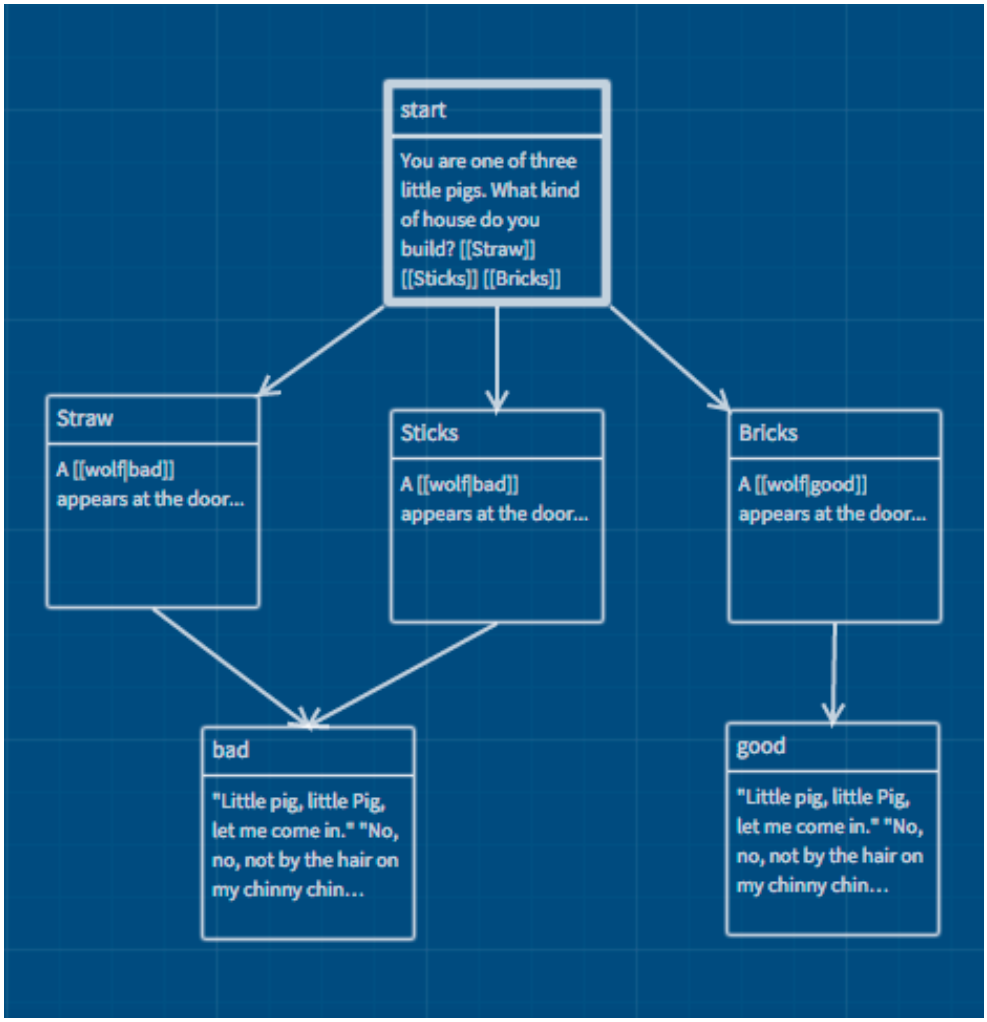
2.4 Alternative and Simultaneous Story Structure

Linear logic provides the tools we need to specify story worlds in such a way that we can define relationships of interest between story events, but there are still quite a few choices about which components to model in the story world that give rise to different event relationships. Let us now be more specific about what kinds of event relationships we would like to aim to model, and explain how we can use linear logic to model them.

2.4.1 Alternative Storylines

One kind of narrative structure of interest to interactive storytelling is *alternative storylines*. This structure is present in any Choose Your Own Adventure or Twine game: a reader is presented with multiple choices and, upon selecting one, is cut off from the other choices until replay (or until that scene is presented again, in the case of cyclic stories). These stories have a natural representation as a *graph*, with story scenes (or *passages* in Twine parlance) modeled by graph vertices, and possible consequences y of a scene vertex x modeled as directed edges from x to y. Sometimes this kind of narrative structure is called *branching*, but by calling attention to *alternatives* rather than branching, we aim to include story graphs that are not simply trees: paths that diverge (branch) may converge in a later scene, and divergence is not given any priority over convergence.

The graph structure just described is depicted in the Twine editor for the sake of the author to visualize their story structure. To relate this idea to our example, we can render the Three Little Pigs fabula as a Twine story in which the choices represent the alternatives between building a house out of straw, sticks, or bricks:



This depiction of the story structure can be seen as containing three alternative storylines, each corresponding to one "playthrough" of the game. Two of those playthroughs end in the "bad" passage in which the wolf blows down the house, and the third ends in the "good" passage in which the wolf cannot blow down the house. The story graph is a more compact representation of these three stories that makes use of the shared structure between the first and second story, as well as their shared beginning. It can be understood as a *potential narrative* because each interaction with a player generates one of the three stories.

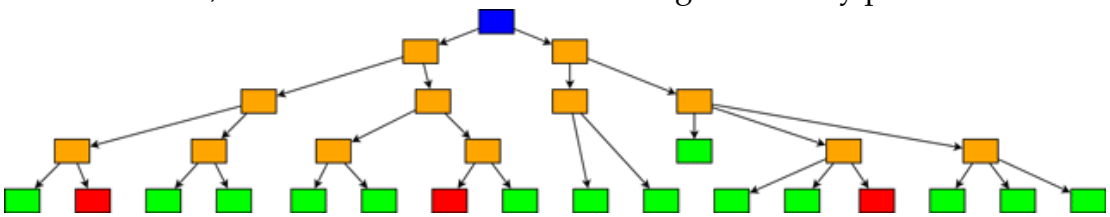
Alternative structure in stories already yields many interesting authoring choices. In an online article, Sam Kabo Ashwell reviews several patterns in the edges-as-alternatives graph structure for interactive storytelling, such as those in Figure 2.2. [3]

However, many rich interactive storytelling examples make use of complex state tracking, such as inventories in parser interactive fiction. These graphs do not depict how the story will vary depending on that state.

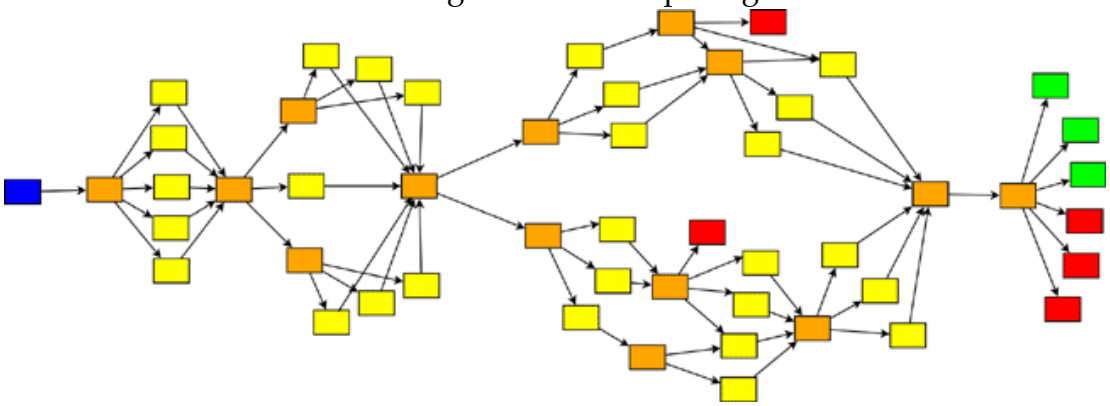
[3]The article is available at
`https://heterogenoustasks.wordpress.com/2015/01/26/standard-patterns-in-choice-based-games/`.

Figure 2.2: Four structural patterns in branching narrative.
Diagrams created by Sam Kabo Ashwell, used with permission.
Color key: red nodes are endings representing failure; green nodes are endings representing success; blue nodes are the start of the story; orange nodes are choice points; yellow nodes are waypoints (they only have a single out-edge).

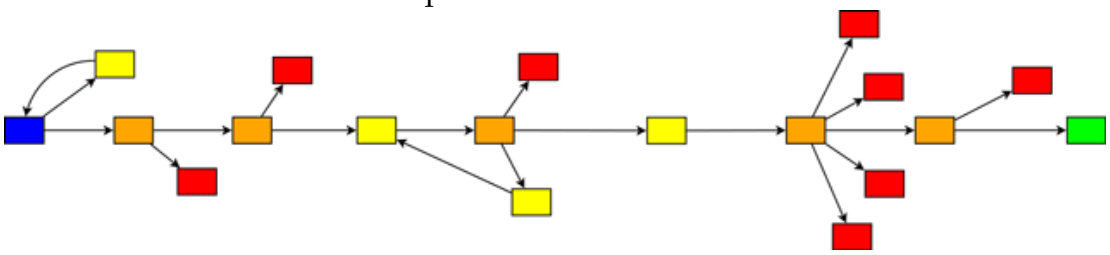
The *time cave*, in which branches never converge and every path is distinct:



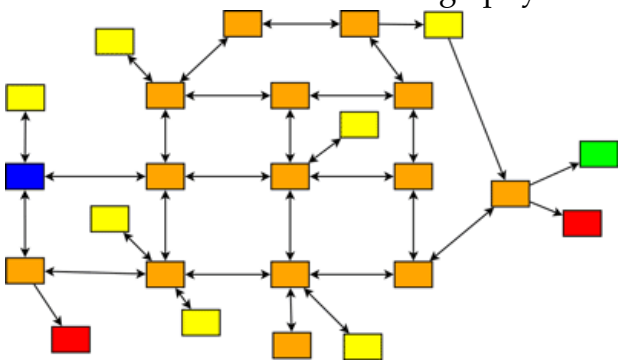
The *branch-and-bottleneck*, in which outward-branching alternatives occasionally re-converge on common passages:



The *gauntlet*, in which one particular path serves as the backbone for the story, and all other paths lead to dead ends:



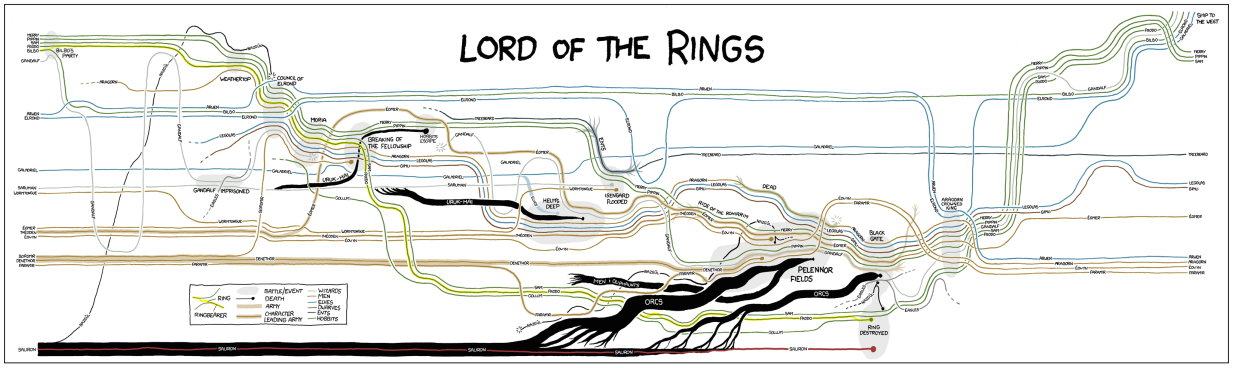
The *open map*, in which symmetric connections are made between nodes, often used to simulate reversible movement through physical space:



2.4.2 Simultaneous Storylines

Alternative structures are one way to conceptualize a story in terms of constituent pieces, but they are not the only way. Rather than imagining making choices from the perspective of one of the three little pigs, we might like to model all of the characters' actions and interactions together. We can map a narrative onto a graph in a different way: edges are characters in certain states of the narrative, and nodes are scenes where those characters interact.

In one strip of the online comic XKCD, Randall Munroe depicts several popular movie plots in this style, such as this plot graph for The Lord of the Rings: [4]

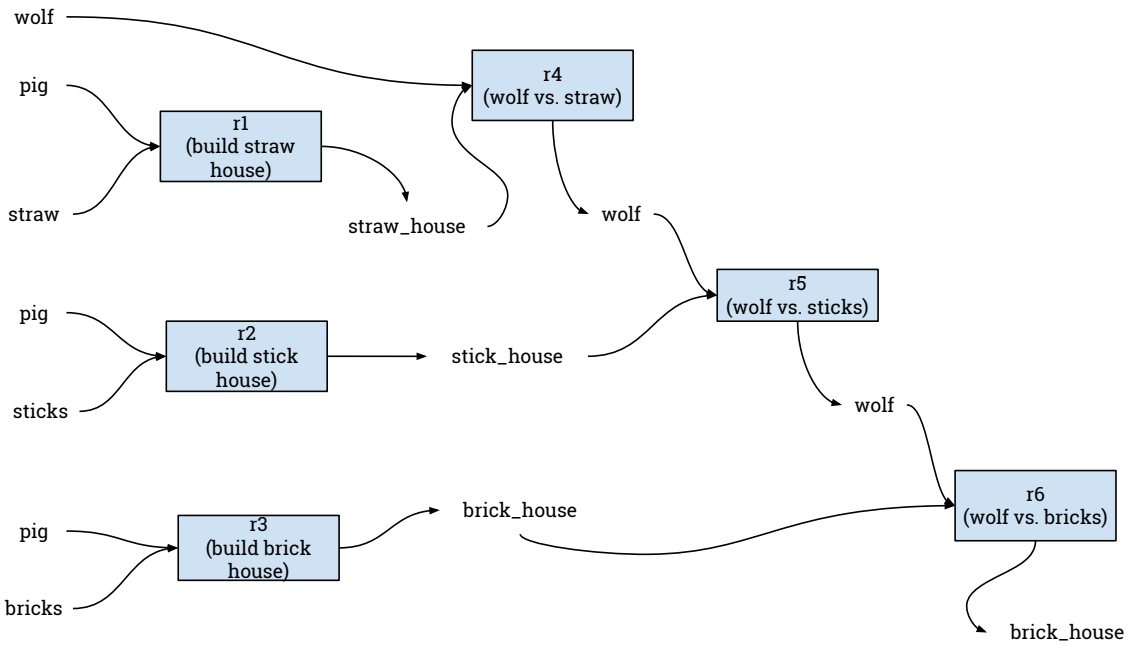


Along the vertical axis, characters are placed near one another to represent sharing a scene or far apart to represent independent activity. The horizontal axis is the forward progression of time. The *ring*, an imporant narrative resource, is drawn as an overlay atop the character holding it.

This version of narrative structure lends itself to the *analysis* of a plot more readily than alternative structure: we can see, for instance, that climactic moments correspond to the convergence of a large number of characters, and we can trace the thread of a particular character or group of characters to examine their overall influence on the plot. More relevantly to narrative play, we can envision the story in terms of autonomous, interacting characters who might have independent motives, interiority, and relationships.

To model the Three Little Pigs narrative in this way, we need to break down our story into smaller component pieces: one for each pig and the wolf, and perhaps also the building materials they use for their houses. If we are more precise than Munroe's drawing about what exactly a "scene" does, in terms of how it transforms these narrative resources, then we can draw the story's simultaneous structure as follows:

[4] Larger version can be found at `https://xkcd.com/657/`.



This diagram represents something more *static* than the Twine game version: it does not present clear affordances for interaction. However, it is also more *systematic*, in that it forces us to imagine a world model in which the events in the story could logically and physically take place. What we present in the next sections is a formal setting in which *both* of these valuable aspects of structural expression are available.

2.4.3 Alternatives and Simultaneity in Linear Logic

Linear logic's resource-oriented mechanisms enables connectives that neatly map onto alternative ($A \& B$) and simultaneity ($A \otimes B$). Due to the "use once" principle of assumptions in linear logic, these manifest as two forms of logical conjunction ("and"). The first one of these we will explain is $A \otimes B$, which means that resources A and B are available simultaneously.

The $\otimes$ and $\multimap$ connectives together give us the ability to write propositions such as $\mathsf{pig} \otimes \mathsf{bricks} \multimap \mathsf{brick\_house}$, modeling a resource exchange that characterizes the "pig builds a brick house" scene in The Three Little Pigs. Here, the $\otimes$ connective serves to conjoin the two atomic pig and bricks propositions into a more complex proposition. There is a *unit* of the $\otimes$ connective indicating the null resource, spelled 1.[5]

The other form of conjunction, $A \& B$, means that a choice between A and B is avail-

[5]For 1 to be a unit of $\otimes$ means that, for any A, $A \otimes 1$ is equivalent to (interprovable with) A and $1 \otimes A$.

able. This connective lets us model the first passage in our Twine game representation of The Three Little Pigs as $\mathsf{pig} \multimap \mathsf{brick\_house} \& \mathsf{straw\_house} \& \mathsf{stick\_house}$. [6]

To model stories in linear logic, we map atomic propositions onto pieces of narrative state. In the case of purely alternative story structures, a single piece of narrative state representing each scene/passage will suffice. In the case of narratives with simultaneous structure, propositions will model potentially more complex narrative elements and their relationships.

Here is a model of the additive story world for Three Little Pigs:

Let $\Delta =$

$$\{\mathsf{pig} \multimap (\mathsf{straw\_house} \,\&\, \mathsf{stick\_house} \,\&\, \mathsf{brick\_house}),$$
$$\mathsf{straw\_house} \multimap \mathsf{wolf\_wins},$$
$$\mathsf{stick\_house} \multimap \mathsf{wolf\_wins},$$
$$\mathsf{brick\_house} \multimap \mathsf{pig\_wins},$$
$$\mathsf{pig}\}$$

Stories will be proofs of the sequent

$$\Delta \vdash \mathsf{wolf\_wins}$$

or

$$\Delta \vdash \mathsf{pig\_wins}$$

Here is a model of the simultaneous story world, in which we make use of linear logic's notion of simultaneity ($\otimes$) to introduce three copies of the *pig* narrative resource, and we also represent finer-grained narrative resources such as the building materials for the pigs' houses:

Let $\Delta =$

$$\{\mathsf{pig} \otimes \mathsf{straw} \multimap \mathsf{straw\_house},$$
$$\mathsf{pig} \otimes \mathsf{sticks} \multimap \mathsf{stick\_house},$$
$$\mathsf{pig} \otimes \mathsf{bricks} \multimap \mathsf{brick\_house},$$
$$\mathsf{wolf} \otimes \mathsf{straw\_house} \multimap \mathsf{wolf},$$
$$\mathsf{wolf} \otimes \mathsf{stick\_house} \multimap \mathsf{wolf},$$
$$\mathsf{wolf} \otimes \mathsf{brick\_house} \multimap \mathsf{brick\_house},$$
$$\mathsf{wolf}, \mathsf{pig}, \mathsf{pig}, \mathsf{pig}, \mathsf{straw}, \mathsf{sticks}, \mathsf{bricks}\}$$

Stories will be proofs of the sequent

[6] There is a unit of $\&$ as well, spelled $\top$, which we do not include here because it is not needed for our examples.

$$\Delta \vdash \mathsf{brick\_house}$$

Note that in both of these models, scenes (represented as propositions of the form $A \multimap B$) appear on the same level as atomic narrative resources like pig. This means that the logic will enforce a *use exactly once* semantics for those scenes, embodying a kind of *narrative drive* (we are required to include those scenes to prove the sequent). If we want to make a scene A optional, all we need to do is create a choice between that scene and the null resource: $A \& 1$. Persistence ($!A$) can be used to model stories where scenes may repeat an arbitrary number of times.

2.4.4 Proofs as Stories

Finally, we can present concretely the mapping of stories onto proofs. For the Twine-based version of the story, where story events are related as alternatives, we can model the story world with $\Gamma =$

$$\begin{array}{rcl} r_1 & : & \mathsf{pig} \multimap (\mathsf{straw\_house} \,\&\, \mathsf{stick\_house} \,\&\, \mathsf{brick\_house}) \\ r_2 & : & \mathsf{straw\_house} \multimap \mathsf{wolf\_wins} \\ r_3 & : & \mathsf{stick\_house} \multimap \mathsf{wolf\_wins} \\ r_4 & : & \mathsf{brick\_house} \multimap \mathsf{pig\_wins} \end{array}$$

The initial state Δ_0 will be the single resource pig, corresponding to the initial Twine passage.

Here is a derivation corresponding to the story of the pig that builds the straw house in the additive story world:

$$\dfrac{\dfrac{}{\mathsf{pig} \vdash \mathsf{pig}} \quad \dfrac{\dfrac{\dfrac{}{\mathsf{straw\_house} \vdash \mathsf{straw\_house}} \quad \dfrac{}{\mathsf{wolf\_wins} \vdash \mathsf{wolf\_wins}} \multimap L(r_2)}{\mathsf{straw\_house} \vdash \mathsf{wolf\_wins}}}{\mathsf{straw\_house} \,\&\, \mathsf{stick\_house} \,\&\, \mathsf{brick\_house} \vdash \mathsf{wolf\_wins}} \,\&L_1}{\Gamma; \mathsf{pig} \vdash \mathsf{wolf\_wins}} \multimap L(r_1)$$

The above derivation can also be thought of as a proof that the wolf can win (against a pig who makes a straw or stick house). Below is a proof that the pig can win , representing the story in which the pig builds the brick house:

$$\dfrac{\dfrac{}{\mathsf{pig} \vdash \mathsf{pig}} \quad \dfrac{\dfrac{\dfrac{}{\mathsf{brick\_house} \vdash \mathsf{brick\_house}} \quad \dfrac{}{\mathsf{pig\_wins} \vdash \mathsf{pig\_wins}} \multimap L(r_4)}{\mathsf{brick\_house} \vdash \mathsf{pig\_wins}}}{\mathsf{straw\_house} \,\&\, \mathsf{stick\_house} \,\&\, \mathsf{brick\_house} \vdash \mathsf{pig\_wins}} \,\&L_3}{\mathsf{pig} \vdash \mathsf{pig\_wins}} \multimap L(r_1)$$

Every different proof with $\Gamma; \Delta_0 \vdash \gamma$ at its root corresponds to a different potential narrative embodied by the alternative structure.

To describe *simultaneous* relationships between story events, we need to divide up the monolithic state of "which Twine passage are we in" into smaller pieces, such as

each pig, the wolf, the building materials the pigs use to make their houses, and the completed houses.

The story ending is now not a a single fact (the wolf or the pig winning) but rather some composition (specifically, a $\otimes$ conjunction) of outcomes for different narrative resources.

In this version of the story, $\Gamma =$

$$\begin{aligned}
r_1 &: \mathsf{pig} \otimes \mathsf{straw} \multimap \mathsf{straw_house} \\
r_2 &: \mathsf{pig} \otimes \mathsf{sticks} \multimap \mathsf{stick_house} \\
r_3 &: \mathsf{pig} \otimes \mathsf{bricks} \multimap \mathsf{brick_house} \\
r_4 &: \mathsf{wolf} \otimes \mathsf{straw_house} \multimap \mathsf{wolf} \\
r_5 &: \mathsf{wolf} \otimes \mathsf{stick_house} \multimap \mathsf{wolf} \\
r_6 &: \mathsf{wolf} \otimes \mathsf{brick_house} \multimap \mathsf{brick_house}
\end{aligned}$$

The initial configuration for the story is the state

$$\Delta_0 = \{\mathsf{pig}, \mathsf{pig}, \mathsf{pig}, \mathsf{straw}, \mathsf{bricks}, \mathsf{sticks}, \mathsf{wolf}\}$$

And we can create proofs of the sequent

$$\Delta_0 \vdash \mathsf{brick\_house}$$

to represent stories ending with just the brick house standing, i.e. the canonical Three Little Pigs folktale ending. Here is a proof that the brick house can be the only one left standing, representing the canonical Three Little Pigs story:

Let $\mathcal{D}_1 =$

$$\frac{\mathsf{pig} \vdash \mathsf{pig} \quad \mathsf{bricks} \vdash \mathsf{bricks}}{\mathsf{pig}, \mathsf{bricks} \vdash \mathsf{pig} \otimes \mathsf{bricks}} \otimes R$$

Let $\mathcal{D}_2 =$

$$\frac{\overline{\mathsf{pig} \vdash \mathsf{pig}} \quad \overline{\mathsf{straw} \vdash \mathsf{straw}}}{\mathsf{pig}, \mathsf{straw} \vdash \mathsf{pig} \otimes \mathsf{straw}} \otimes R$$

Let $\mathcal{D}_3 =$

$$\frac{\overline{\mathsf{pig} \vdash \mathsf{pig}} \quad \overline{\mathsf{sticks} \vdash \mathsf{sticks}}}{\mathsf{pig}, \mathsf{sticks} \vdash \mathsf{pig} \otimes \mathsf{sticks}} \otimes R$$

Let $\mathcal{D}_4 =$

$$\frac{\overline{\mathsf{wolf} \vdash \mathsf{wolf}} \quad \overline{\mathsf{straw\_house} \vdash \mathsf{straw\_house}}}{\mathsf{wolf}, \mathsf{straw\_house} \vdash \mathsf{wolf} \otimes \mathsf{straw\_house}} \otimes R$$

Let $\mathcal{D}_5 =$

$$\frac{\overline{\mathsf{wolf} \vdash \mathsf{wolf}} \quad \overline{\mathsf{stick\_house} \vdash \mathsf{stick\_house}}}{\mathsf{wolf}, \mathsf{stick\_house} \vdash \mathsf{wolf} \otimes \mathsf{stick\_house}} \otimes R$$

in

$$\dfrac{\mathcal{D}_1 \quad \dfrac{\mathcal{D}_2 \quad \dfrac{\mathcal{D}_3 \quad \dfrac{\mathcal{D}_4 \quad \dfrac{\mathcal{D}_5 \quad \dfrac{\dfrac{}{\mathsf{wolf} \vdash \mathsf{wolf}} \quad \dfrac{}{\mathsf{brick\_house} \vdash \mathsf{brick\_house}}}{\mathsf{brick\_house}, \mathsf{wolf} \vdash \mathsf{wolf} \otimes \mathsf{brick\_house}} \quad \dfrac{}{\mathsf{brick\_house} \vdash \mathsf{brick\_house}}}{\mathsf{brick\_house}, \mathsf{wolf} \vdash \mathsf{brick\_house}} \multimap L(r_6)}{\mathsf{stick\_house}, \mathsf{brick\_house}, \mathsf{wolf} \vdash \mathsf{brick\_house}} \multimap L(r_5)}{\mathsf{stick\_house}, \mathsf{straw\_house}, \mathsf{brick\_house}, \mathsf{wolf} \vdash \mathsf{brick\_house}} \multimap L(r_4)}{\mathsf{straw\_house}, \mathsf{brick\_house}, \mathsf{pig}, \mathsf{sticks}, \mathsf{wolf} \vdash \mathsf{brick\_house}} \multimap L(r_2)}{\mathsf{brick\_house}, \mathsf{pig}, \mathsf{pig}, \mathsf{straw}, \mathsf{sticks}, \mathsf{wolf} \vdash \mathsf{brick\_house}} \multimap L(r_1)}{\Delta_0 \vdash \mathsf{brick\_house}} \multimap L(r_3)$$

Note the lopsided structure of the above proof: we only apply $\multimap L$ rules up the entire tree except to derive rules' premises from the context. The proof could almost be read as a sequence of rule applications $r_3; r_1; r_2; r_4; r_5; r_6$ that navigate from the bottom sequent (in which the context represents the initial story configuration) to the top-right sequent (in which the context represents the final story configuration). On the other hand, by tracking exactly *which* resources in the context are consumed and produced by each rule application, we can extract a partial ordering between these rule applications, yielding the diagram depicted in Section 2.4.2. The rule sequence plus its input-output dependency links with other rules is exactly the information captured in the formal proof term syntax that we interpret as stories in Chapter 3.

Another note about the above proof is that it is not the only one that may follow from the story world rules and initial configuration $\Gamma; \Delta_0$. We can derive different outcomes where two or three of the houses are left standing, i.e.

$$\Delta_0 \vdash \mathsf{brick\_house} \otimes \mathsf{stick\_house}$$

and

$$\Delta_0 \vdash \mathsf{brick\_house} \otimes \mathsf{stick\_house} \otimes \mathsf{straw\_house}$$

That these sequents are provable suggests that even in our "simultaneous" story world, there is some alternative structure to consider—i.e. there is nondeterminism in the system of inference rules, and several might apply.

To see what stories these unconventional endings might give rise to, let us examine a proof of the sequent $\Delta \vdash \mathsf{brick\_house} \otimes \mathsf{straw\_house} \otimes \mathsf{stick\_house}$ (where we abbreviate the right-hand side of the sequent A).

Let $\mathcal{D}_6 =$

$$\dfrac{\dfrac{}{\mathsf{brick\_house} \vdash \mathsf{brick\_house}} \quad \dfrac{\dfrac{}{\mathsf{straw\_house} \vdash \mathsf{straw\_house}} \quad \dfrac{}{\mathsf{stick\_house} \vdash \mathsf{stick\_house}}}{\mathsf{straw\_house}, \mathsf{stick\_house} \vdash \mathsf{straw\_house} \otimes \mathsf{stick\_house}} \otimes R}{\mathsf{brick\_house}, \mathsf{straw\_house}, \mathsf{straw\_house} \vdash A} \otimes R$$

in

$$\dfrac{\mathcal{D}_1 \quad \dfrac{\mathcal{D}_2 \quad \dfrac{\mathcal{D}_3 \quad \dfrac{\dfrac{\dfrac{}{\mathsf{wolf} \vdash \mathsf{wolf}} \quad \dfrac{}{\mathsf{brick\_house} \vdash \mathsf{brick\_house}}}{\mathsf{brick\_house}, \mathsf{wolf} \vdash \mathsf{wolf} \otimes \mathsf{brick\_house}} \otimes R \quad \mathcal{D}_6}{\mathsf{straw\_house}, \mathsf{brick\_house}, \mathsf{stick\_house}, \mathsf{wolf} \vdash A} \multimap L(r_6)}{\mathsf{straw\_house}, \mathsf{brick\_house}, \mathsf{pig}, \mathsf{sticks}, \mathsf{wolf} \vdash A} \multimap L(r_2)}{\mathsf{brick\_house}, \mathsf{pig}, \mathsf{pig}, \mathsf{straw}, \mathsf{sticks}, \mathsf{wolf} \vdash A} \multimap L(r_1)}{\Delta \vdash A} \multimap L(r_3)$$

Again reading the named rules used in the proof from bottom to top, we can see we have a shorter story wherein after each pig builds their house, the wolf simply visits the brick house first and is eliminated from the narrative state there, leaving all three houses standing.

Some computational narrative researchers would consider such a story to be poorly formed, particularly according to popular Western story aesthetics: it can hardly be argued that this narrative has any *conflict*. Furthermore, two characters in the story (the straw house and brick house pig) are seemingly only referred to once, when they build their houses, and never interact with the narrative again, making their role in the story seem inessential. Riedl and Young [RY04] have investigated means of manipulating similar inference systems (planning) to introduce conflict by way of individual character goals that compete over shared resources. In future work we would like to investigate the applicability of their techniques to linear logic.

2.5 First-Order Linear Logic

One advantage of working with the logical methodology we have chosen is that it is easy to enrich the logic with orthogonal constructs. For instance, we can enrich our logic of atomic propositions to allow us to state more complex relationships between entities, such as $\mathsf{at\ pig\ straw\_house}$ to represent location or $\mathsf{has\ pig\ straw}$ to represent possession. If we additionally introduce a *quantifier* connective $\forall$ ("for all"), we can write a proposition modeling a *parametric* action where a pig in *any* location L where there is a building material M can be replaced by the pig possessing M:

$$\forall L, M.\mathsf{at\ pig}\ L \otimes \mathsf{at}\ M\ L \ \multimap\ \mathsf{has\ pig}\ M$$

For modeling pre-established narratives like The Three Little Pigs, modeling the story world at this level of detail and parametricity seems unnecessary. However, for the invention of more flexible story worlds prone to expressive collaboration from a human interactor, parametric rules like this one become essential.

The creators of social physics systems like Prom Week emphasize that creating a wide possibility space for narrative unfoldings is essential to the player's discovery of "emergent solutions and surprising, yet satisfying, outcomes." [MTS+11] Expressing the kinds of rules that make up Prom Week's social physics engine (Comme il Faut [MTS+10]) requires that they not be specialized to any particular character in the

story but depend only on certain properties of characters and their relationships. The first-order extension of linear logic enables us to express rules of this form.

Most prior work on the use of linear logic to model stories and games focus *only* on the non-parametric, or *propositional*, fragment of the logic (see Section 2.6.2), and so its range of considered examples has mostly been limited to hand-authored, branching narratives. In Chapter 3, we will show how first-order linear logic scales to account for a richer narrative possibility space. We note that, as a side-effect, we will be able to use the same formalism to describe Twine games, command-line ("parser") interactive fiction, and multi-agent social models like Comme il Faut, and compare the narrative structures that arise from them.

2.6 Related Work

Bosser et al. [BCFC11] similarly identify structures in stories that can be identified as simultaneous and alternative, also modeled in linear lofic. In their case, simultaneous structure is represented by proof terms that record uses of the $\otimes R$ rule as "stories in parallel." However, such proofs still sequentialize story events that we identify as formally independent from one another. Our notion of independence aligns better with the permutability of proof rules than with the commutative structure of the linear context.

We identify two other formalisms that are very closely related to linear logic for story formalization: planning and Petri nets.

2.6.1 Planning

Interactive storytelling research communities have historically used planners (such as STRIPS [FN71] and IPOCL [YPM94]) to model and generate stories [You99, CMC01, RY10]. The planning approach to modeling actions, i.e. by designating facts as *preconditions*, *deleted* by the action, and *added* by the action, has a great deal in common with modeling actions as linear logic implications. In fact, Masseron et al. [Mas93] have shown how linear logic can be used to support planning and equate a proof to a plan. Others have demonstrated the use of linear logic for specific problems thought of as typical planning problems [DSB09, DST09].

So, in the interactive storytelling domain, linear logic can effectively be seen as a logic-based alternative to planning. We see the advantages to a basis in logic as twofold: first, that a long tradition of logics connecting to human epistemology means a wide range of epistemological extensions are available, and there is vast precedent for designing logics so that their different judgments (and corresponding connectives) may fit together, composing into richer formalisms. For example, this precedent is what allows us to seamlessly extend propositional linear logic with first-order quantification. Another, less-explored example is combining linear logic's resource orientation with logical connectives for *knowledge* of specific actors [GBB$^+$06], which we hypothesize could be quite fruitful for narrative modeling.

There are a few more technical differences between the expressivity of linear logic and planning as formalisms. For instance, in linear logic, the *multiplicity* of a fact matters: we cannot write brick_house $\multimap$ wolf to represent "deleting" the brick_house fact in the same way we can in planning, because there may be other copies of the brick_house resource available that we do not account for. (We would consider these statements equivalent, though, when an *invariant* of the program includes having exactly zero or one of a certain resource.) A consequence of this fact is that linear logic *can* refer to multiple instances of something (like money) without propositions indexed by a natural number, while planning cannot.

Another (related) difference is that fact deletion in planning does not require that the fact being deleted is actually *present* for the rule to fire. In linear logic, all "deleted" facts are also "preconditions." Finally, planners typically support *negation* of facts as preconditions for actions. Both of these constructs (unconditional deletion, and negation as failure) are inexpressible in linear logic, and in fact are inconsistent with purely logical characterization in general, because they do not maintain the transitivity of the consequence relation. However, when we consider the extension of our approach into a more practical programming language, we do add support for similar idioms (see Chapter 4).

2.6.2 Petri Nets

Petri nets [Mur89] are a formalism studied in the 1970s primarily for concurrent computation. They are easily interpreted in a graphical languages of nodes and arcs, where nodes may be either *places* (represented with a circle) or *transitions* (represented as a rectangle), where arcs connect places to transitions and transtions to place. Each place may have zero or more *markings* on it.

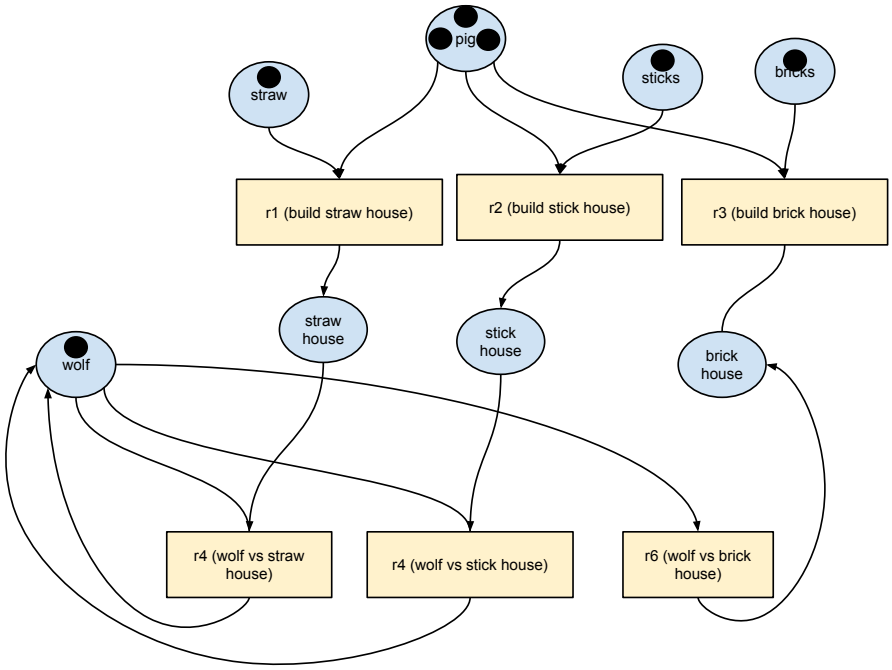
A given Petri net and configuration of markings may evolve to a new configuration as follows: if all of the places that point to (have an arc to) a given transition are marked, the transiton may *fire*, or remove one marking from each input place and add a marking to each output place. Places may have arcs to multiple transitions, suggesting nondeterminism in the system.

Petri nets may be understood as a restricted fragment of linear logic: a Petri net place is modeled by an atomic linear logic proposition, and a Petri net transition matches up to a linear logic implication of the form $!(a_1 \otimes \ldots \otimes a_n \multimap b_1 \otimes \ldots \otimes b_m)$, where each a_i is an input place to the transition and b_is are output places. Every rule must be persistent (thus the ! in front) so that it can model a transition that may fire arbitrarily many times.

The *reachability* problem in Petri nets—whether it is possible for a given configuration to evolve through arbitrarily many rule firings to another configuration—was shown equivalent to provability in this fragment of linear logic by Kanovich. [Kan95]

Note that this correspondence does not take into account the first-order extension of linear logic. For some first-order programs, it may be possible to *ground* them, i.e. give an equivalent propositional program, by instantiating every rule with every possible term given in the domain (assuming the domain is finite, which it may not be), but even when this is possible it results in a combinatorial explosion of the state space. This may

Figure 2.3: Petri net for the simultaneous Three Little Pigs story world.



make Petri nets as a formalism less scalable than first-order linear logic as an authoring tool.

Despite this limit in expressiveness, quite a few applications of Petri nets to games and interactive storytelling have been explored: first, Vega et al. investigate their use in game design [VGN04]; Araújo and Licínio use them to model game mechanics [AR09]; Dang et al. use the equivalent fragment of linear logic to validate interactive story scenarios [DHCS11, DCA13]; and my coauthors and I use that same fragment for generation of story variants [MBFC13] on the fabula of Flaubert's novel *Madame Bovary* [Fla01]. The main *benefit* of the limited expressiveness is that it is possible to do exhaustive analysis, since proof search on that fragment (i.e. reachability for Petri nets) is decidable. [May84]

Petri nets also have the advantage of enjoying a direct visual representation that nicely illustrates the relationship between alternative and simultaneous structure. The Petri net in Figure 2.3 illustrates the Three Little Pigs story world given in Section 2.4.2, and to find potential alternative structure we simply look for places (circles) with multiple out-edges.

Finally, we note that a Petri net-inspired system of markings and transitions, called Machinations, has been built as a visual game design tool [Dor11]. Later work formalized the core of this system and its semantics ("micro-machinations") [VRD14] to find that they differ substantially from Petri nets, losing the correspondence with linear logic.

2.7 Conclusion

We have presented linear logic as a formalism that supports two kinds of structure in narrative fabula, namely *alternatives* and *simultaneity*. We have built upon prior narrative modeling methodologies to map entities, relationships, states, events, stories, and story outcomes onto corresponding logical and proof-theoretic notions, completing the conceptual basis on which the remainder of this thesis builds.

In particular, we have demonstrated several core properties of narrative structure that correspond to the fundamental mechanisms of linear logic. Bosser et al. [BCC10] point out three such characteristics of computational stories: ***generativity***, or the derivability of a variety of different story unfoldings given the same beginning and ending conditions; ***variability***, or the open world assumption characterized by the story rules' independence from the initial story configuration; and ***narrative drive***, or the enforcement of certain story rules (scenes) being used in the proof. These properties emerge from the logic in the form of a nondeterministic inference system, the *hypothetical judgment* (i.e. presence of the context Δ as an essential part of sequents), and the absence of *weakening* making it possible to enforce the usage of assumptions.

We additionally observe the correspondence between linear logic connectives and *alternative* and *simultaneous* relationships between story events. Our observation of the logic's ability to model alternatives is effectively the same as the *variability* property mentioned above, but the *simultaneity* observation was mainly portrayed as causality in prior work. We posit that simultaneity offers a more operational reading of the narrative structure, in which we can imagine several moving parts to a complex interactive narrative, whose independent evolutions compose into a coherent story. Many works of interactive fiction already use such ideas by incorporating persistent state (such as player inventory and tracking of game history), but the simple alternative models of these fictions (e.g. Ashwell's diagrams in Section 2.4) do not depict the more complex story variability that arises from them. Reed and Garbe [RGWFM14] introduce the term *combinatorial narrative* to suggest centralizing the idea of complex narrative states in interactive story authoring, and we posit that linear logic would make a good candidate for modeling and reasoning about those works.

In the next chapter, we will show how to put computers to use on the mathematical formalism presented here in the form of *logic programming* to extract a computational interpretation of story modeling in linear logic. Namely, we extend our mapping to account for story world (or potential narrative) as a program that, when run, generates coherent stories according to the authored rules.