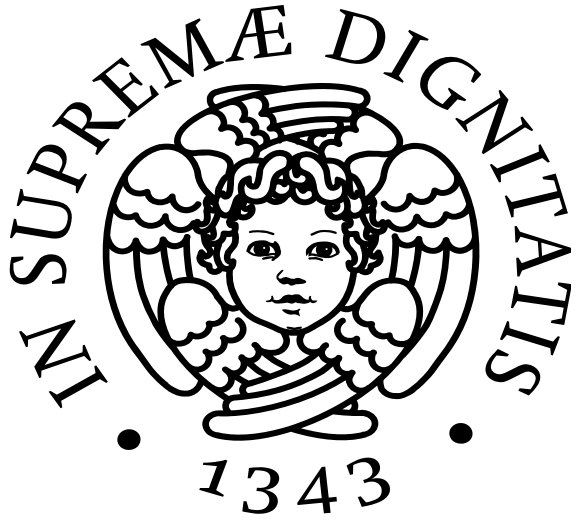
UNIVERSITA` DEGLI STUDI DI PISA

Dipartimento di Filologia, Letteratura Linguistica
CDL Informatica Umanistica
Laurea Magistrale



VERIFICA DI PROPRIETÁ DI MODELLI DI GIOCHI CON TECNICHE DI MODEL CHECKING PROBABILISTICO

RELATORE Prof. Paolo MILAZZO

CORELATORE Dott. Pasquale BOVE

Candidato Dario SESTINI

ANNO ACCADEMICO 2014-15

Riassunto

Il gioco e l'attività di giocare sono insite nell'essere umano il quale mette da sempre alla prova le sue capacità in sfide di ogni sorta. Nei primi due capitoli di questo elaborato si fornisce un quadro teorico e introduttivo al mondo del gioco, analizzando alcuni contributi provenienti dall'area umanistica e dal cerchio un po' più ristretto dei *Games Studies,* i quali recentemente stanno vivendo un momento di felice espansione. Si propone una definizione di gioco partendo dalle definizioni che altri studiosi del fenomeno hanno fornito in passato per poi passare all'analisi del gioco inteso come artefatto, oggetto immaginario o tangibile in grado di far vivere ai giocatori esperienze più o meno significative. Soffermandoci in particolare sui giochi da tavolo, oggetto specifico dell'elaborato, si fa un elenco di alcune delle proprietà che un gioco possiede o dovrebbe possedere al fine di poter offrire ai giocatori un'esperienza di un certo tipo. La progettazione degli artefatti ludici, che prende il nome di *game design*, sarà affrontata a partire da alcuni concetti chiave che ne stanno alla base.
Nel terzo capitolo si affronta in modo più approfondito la modellazione di un gioco e l'analogia che esiste tra questo e i sistemi transizionali, per poi fornire un esempio di studio di un modello di gioco implementato attraverso la piattaforma software PRISM. In questo capitolo si offre inoltre una breve panoramica sul model checking, mettendo in luce alcune delle problematiche legate ai metodi formali di verifica e i risultati che a oggi si è riusciti a ottenere in questo campo.
Infine l'ultimo capitolo contiene tre casi di studio con i quali si intende esplorare concretamente una metodologia con la quale, attraverso la verifica delle proprietà di un gioco con tecniche di model checking probabilistico e statistico, si possa rendere più efficace la progettazione di artefatti ludici di qualsiasi genere. In particolare nei casi di studio verranno mostrati alcuni test effettuati su modelli di gioco ispirati a giochi realmente esistenti allo scopo di verificare formalmente alcune delle proprietà viste nei capitoli precedenti.

INDICE

TITOLO:

VERIFICA DI PROPRIETÁ DI MODELLI DI GIOCHI CON TECNICHE DI MODEL CHECKING PROBABILISTICO

1 INTRODUZIONE

Le tecniche di modellazione e di model checking [1] sono state applicate nel corso degli anni a svariati campi del sapere rendendosi sempre più utili sia in fase di progettazione di un dato sistema, sia nella risoluzione delle criticità che questo può presentare una volta sviluppato. Si è scelto in questo lavoro di utilizzare alcune di queste tecniche nell'ambito del game design [2], disciplina che mira al perfezionamento delle metodologie di progettazione di artefatti ludici. Nel tentativo di definire un metodo in grado di aiutare il progettista a compiere scelte di design in modo più efficace, si è inteso sfruttare nei tre casi di studio l'analogia che esiste tra questi e i sistemi transizionali in cui gli stati e le transizioni possono rappresentare rispettivamente le possibili configurazioni e i modi in cui il sistema è in grado di raggiungerle. L'analogia ci ha permesso, tramite un linguaggio *state-based*, di descrivere modelli di giochi sui quali poter verificare determinate proprietà attraverso l'applicazione di tecniche di model checking sia probabilistico che statistico: il software che si è scelto di utilizzare è PRISM Model Checker [3] il quale è in grado di supportare modelli come *Markov chains* o *Markov decision process*. Rappresentando quindi in modo astratto le istanze in gioco e descrivendone l'interazione sottoforma di relazioni di transizione è stato possibile quantificare la probabilità di un determinato evento su tutte le partite giocabili (model checking probabilistico), quando le dimensioni del modello lo hanno permesso, o su un ampio *sample* casualmente scelto nell'insieme dei posibili cammini (model checking statistico). Dei tre modelli implementati il primo è il modello del Gioco dell'Oca con il quale si è voluto osservare gli effetti dell'inserimento di un *malus* all'interno di una dinamica fortemente aleatoria; il secondo è il modello di un gioco di strategia ispirato al famoso Risiko [4], il quale è a sua volta caratterizzato da una componente casuale dovuta alle meccaniche di gestione dei conflitti e di acquisizione delle risorse, tramite il quale si è inteso simulare alcuni dei possibili approcci alla partita al fine di verificare l'esistenza di possibili strategie dominanti; infine il terzo è il modello di un gioco del tipo

dungeons crawl nel quale si è indagato sulle dinamiche legate ai giochi cooperativi e alla collaborazione tra i giocatori. I risultati ottenuti hanno uno scopo dimostrativo e sono da intendere come funzionali all'obiettivo di esplorazione di una possibile metodologia applicabile al mondo della progettazione ludica.

2 IL GIOCO: UN'ATTIVITÁ UMANA

2.1 Una definizione di gioco: atto spontaneo e atto volontario

Nel corso della sua evoluzione sia come individuo che come specie, l'uomo ha da sempre espresso la sua personalità giocando. Le spinte per così dire ancestrali che sottendono al gioco in senso lato sono varie e sono state nei secoli declinate nei modi più diversi. Una definizione di gioco può aiutare a circoscrivere il contesto di questo studio. L'attività del gioco si manifesta nell'uomo fin da subito, in modo spontaneo e naturale. Durante l'infanzia infatti il bambino tende al gioco senza necessitare di spinte esterne o motivazioni di qualche sorta. Il bambino, si può dire, gioca e basta, per il solo piacere che ne ricava. Questa attività però, anche se rimarrà nella sostanza costante nel corso della vita, tende a subire delle trasformazioni che ne modificano i risultati. In [5], Caillois propone innanzitutto una definizione della parola stessa "gioco". Secondo il Caillois infatti la nozione di gioco e, più precisamente l'atto che la reifica, può essere vista come una sestupla di aggettivi. Il gioco è quindi un'attività:

- libera
- separata
- incerta
- improduttiva
- regolata
- fittizia

Libera in quanto l'individuo decide deliberatamente di giocare, non vi è costretto, altrimenti l'essenza del gioco stesso verrebbe a mancare. **Separata** dal contesto nella quale viene svolta, sia sul piano temporale che, possibilmente, su quello dello spazio. È anche un'attività **incerta**, nel senso che non c'è possibilità di stabilirne in anticipo né l'esito né tanto meno lo svolgimento, dal momento

che dipenderanno in buona parte dagli "attori" che di volta in volta vi si dedicheranno. Il gioco è poi sicuramente un'attività **improduttiva** che, come dice Caillois, si verifica al solo scopo di consumarsi in se stessa: alla fine di ogni partita, non importa quale sarà il risultato finale, tutto verrà azzerato per poter ricominciare dal principio, nessun bene e nessuna ricchezza saranno prodotti, ci sarà soltanto "uno spostamento di proprietà all'interno della cerchia dei giocatori" [5]. In questo senso si potrebbe azzardare che il gioco, almeno nelle sue forme più pure, può contenere un germe rivoluzionario che va contro alla tendenza predominante del profitto ad ogni costo. Ovviamente questo germe viene meno se si getta lo sguardo alle manifestazioni reali del gioco in epoca moderna e a un certo tipo di gioco d'azzardo istituzionalizzato, i quali fanno dello sport, per esempio, oltre che occasione di godimento anche e soprattutto fonte di lucro. Per quanto riguarda invece le ultime due tipologie, Caillois ritiene che non possano essere compresenti: "I giochi, dunque, non sono regolati e fittizi. Sono piuttosto o regolati o fittizi". Quindi in definitiva, sono **regolati** e sottoposti a una serie di convenzioni stabilite in partenza che in qualche modo "sospendono le leggi ordinarie [instaurando] una legge nuova che è la sola a contare"; oppure sono **fittizi**, cioè inscritti in una "consapevolezza specifica di una diversa realtà" [5][1].

Nell'analisi del sociologo francese queste caratteristiche descrivono in modo formale la nozione di gioco e lo mettono in rilievo nei confronti di ciò che alla sfera del gioco invece non appartiene. Cambiando il punto di vista e cercando di esaminare le qualità che, al contrario, raggruppano le varie tipologie di gioco in base a una certa originalità impossibile da semplificare ulteriormente, Caillois mette in piedi una suddivisione in quattro diversi quadranti all'interno dei quali è possibile inserire giochi analoghi. I quattro poli separano i giochi a seconda che questi siano guidati principalmente dalla competizione (*agon*), dalla sorte (*alea*), dal simulacro (*mimicry*) o dalla vertigine (*ilinx*). All'interno dei quadranti esiste un'ulteriore possibile suddivisione che segue due

1 Come esempio si può riportare, sempre dal libro di Caillois, i bambini che, per spirito di emulazione, decidono di "giocare a giocare agli scacchi".

opposte direttrici: da una parte si avrà una l'improvvisazione libera e spensierata che viene denominata *paidia*, dall'altra invece una spinta tutta razionale e conscia che prende il nome di *ludus*. Quindi, per ogni categoria di gioco, sarà possibile individuarne le varianti del tutto spontanee, che seguono solamente una spinta vitale esule dal raziocinio e, allo stesso tempo, le varianti declinate in una maniera del tutto arbitraria la quale si regge sulla volontà di cimentarsi in una sfida di qualche tipo, contro un ostacolo astratto o concreto, dotato di autonomia oppure inerte.

Il risultato di una tale classificazione è quindi quello riportato nella tabella qui di seguito [5]:

	AGON (competizione)	ALEA (fortuna)	MIMICRY (simulacro)	ILINX (vertigine)
PAIDIA chiasso agitazione four-rire aquilone solitari cruciverba LUDUS	Corse Combattimenti Atletica (non soggetti a regolamento) Boxe Biliardo Scherma Dama Calcio Scacchi Competizioni Sportive in genere	Testa o croce Filastrocche per fare la conta Scommesse Roulette Lotterie: -semplici -composte -a ripetizione	Imitazioni infatili Giochi illusionistici Bambola Costumi variazioniMaschera Travestimento Teatro Arti dello Spettacolo	Roteare infantile Giostra Altalena Valzer Volador Luna Park Sci Alpinismo Acrobazia

Tabella 2.1: classificazione dei giochi secondo Caillois

2.2 Gioco-giocare, una possibile definizione

Potrebbe essere utile a questo punto tentare di dare una definizione che ci offra un punto di riferimento per quanto riguarda l'attività ludica dell'uomo, in modo da stabilire una base epistemologica comune a tutto lo studio. Esistono varie definizioni possibili che descrivono più o meno approfonditamente un aspetto piuttosto che un altro, ma prima di darne una è bene distinguere due concetti che fino ad ora sono rimasti per così dire "fusi" nello stesso termine, e che potrebbero altrimenti rischiare di generare possibili incomprensioni. Bertolo e Mariani in [2] si soffermano sulla differenza tra due diverse accezioni della parola gioco. In particolare, introducendo un po' quella che è la moderna disciplina dei Game-Studies, precisano come all'interno di quest'ultima la parola *game* possa avere sia il valore di "giocare", esperire un qualcosa tramite una pratica ludica, sia valore di "artefatto", oggetto tangibile o meno che permette l'esperienza ludica in sé. Per il momento ci dedicheremo soltanto al primo dei due significati, ovvero all'atto stesso del giocare a un gioco. In generale, tornando un po' ai concetti utilizzati dal Caillois, l'attività ludica può essere quindi vista come:

- *un atto volontario e in un certo senso improduttivo, di cimentarsi con ostacoli non necessari ottenendo risultati in qualche misura quantificabili*

Per quanto sinteticamente esauriente, questa definizione ha in se una lacuna di fondamentale importanza: non fa nessun riferimento alle regole. Ogni gioco, sia esso progettato per essere giocato da soli o in compagnia, all'aperto o in casa, sul tavolo o su di un prato, ha come condizione preliminare e necessaria che chi intende giocarvi accetti senza riserve il comparto di regole che determina cosa è lecito e cosa non è lecito fare. Si è liberi di scegliere se giocare o meno, ma non si è o non si dovrebbe essere liberi di scegliere se seguire o meno le regole di un gioco una volta che iniziamo a giocare. È possibile invece stabilire delle regole di volta in volta diverse per lo stesso

gioco ma che comunque dovranno essere rispettate per tutta la durata dell'attività ludica.

Per giungere a una definizione esaustiva ci viene in soccorso Bernard Suits, che in [6] fissa tre nodi chiave nella lettura di che cosa è "giocare":

- l'obiettivo ludico;
- le regole;
- l'atteggiamento di chi gioca;

Queste componenti offrono un'impalcatura solida che permette al filosofo di dare una definizione al quanto esauriente. Affinché si possa dare una qualche attività ludica si deve per prima cosa stabilire quale è l'obiettivo del gioco al quale si vuol giocare. Ma cosa succederebbe se, una volta stabilito quale è il goal chi decidesse di partecipare alla sfida pretendesse di farlo a modo suo, senza dare attenzione alle regole: certamente in questo caso non si tratterebbe dell'attività di giocare, ma di qualcosa che esula dal campo di questo studio. È necessario, come detto sopra, che si dia un qualche regolamento, sia esso scritto o verbale, che venga accettato da tutti coloro che intendono partecipare e non solo, perché tutto questo non basterebbe se chi si sta impegnando in un'attività ludica non mantenesse per tutta la durata dell'esperienza un atteggiamento che permetta di tenere vivo il giocare stesso, che faccia sì che non si rompa il "cerchio magico"[2] all'interno del quale i giocatori, interagendo, fanno esperienze ludiche significative. Riprendendo la bozza di definizione data in precedenza e aggiungendo quindi i riferimenti alle regole e all'atteggiamento ludico a cui si è accennato, il giocare diventa:

- *l'atto volontario e improduttivo, di cimentarsi con ostacoli non necessari ottenendo risultati quantificabili, senza mai venir meno al patto che si instaura tra i partecipanti e che prevede il pieno rispetto delle regole del gioco*

A questo punto ci si può ritenere soddisfatti e si può andare a descrivere un po' più in dettaglio quello che abbiamo chiamato cerchio magico. Questo concetto è stato introdotto da J.Huizinga,

2 Il concetto di "cerchio magico" sarà approfondito nei prossimi paragrafi.

studioso olandese autore di [7], una delle opere che hanno in qualche modo contribuito alla nascita dell'interesse scentifico per il mondo del gioco. In questo suo volume Huizinga si sofferma sulla separazione spazio-temporale che solitamente caratterizza in maniera più o meno marcata l'attività ludica, accomunando formalmente quest'ultima a una pratica cultuale o a una rappresentazione teatrale:

> *"Come formalmente non vi è distinzione tra un gioco e un rito* [...] *così formalmente non si distingue il luogo destinato al rito da quello destinato al gioco. L'arena, il tavolino da gioco, il cerchio magico, il tempio, la scena* [...]*, tutti sono per forma e funzione dei luoghi di gioco, cioè spazio delimitato, luoghi segregati, cinti, consacrati sui quali valgono proprie e speciali regole. Sono dei mondi provvisori entro il mondo ordinario*" (Huizinga, [1938] 2002,13)

Ecco cosa si intende per cerchio magico, concetto molto utilizzato nello studio moderno del gioco, e che ci offre un appiglio immediato e chiarificante sulla questione della separatezza dell'attività ludica. Chi gioca sa perfettamente quali sono i confini del proprio agire, dove inizia il "terreno" di gioco e dove finisce. Nel caso di un perimetro tracciato per terra, per esempio in un campo di pallacanestro, è facilmente riconoscibile (non solo per chi gioca, ma anche per chi assiste come spettatore) quali sono le delimitazioni che racchiudono lo spazio di gioco; lo stesso può valere per un gioco da tavola che preveda un tabellone sul quale disporre e muovere le pedine dei giocatori. Il concetto di cerchio magico può però essere esteso e abbracciare non solo la porzione dove fisicamente si svolge l'attività ludica, ma anche l'intero ambiente circostante: il palazzetto nel caso della competizione di pallacanestro, la stanza in cui si sta giocando a Monopoli e così via.

Se prima si è parlato di regole può essere utile approfondire ancora un po' il concetto alla luce di quanto appena esposto. Per Maresa Bertolo le regole di un gioco devono innanzitutto esistere (altrimenti non si tratterebbe più di *ludus*, ma di *paidia*), devono essere comunicate chiaramente e inequivocabilmente a tutti i partecipanti e, come si è detto poco sopra, rispettate. "La regola del

gioco" dice la Bertolo "è puramente facoltativa: non possiamo essere obbligati a giocare; e allo stesso tempo è fatalmente coercitiva una volta che il gioco abbia avuto inizio" [2]. Le regole di un gioco descrivono al giocatore come comportarsi all'interno del cerchio magico affinché l'esperienza di gioco non si interrompa e possa diventare significativa.

2.3 Gioco-artefatto

"Un gioco è un sistema al cui interno i giocatori si impegnano in un conflitto artificiale, ben definito da regole, che porta a un risultato quantificabile." [8]

Questa definizione di gioco come artefatto ludico sembra essere esauriente. Un gioco è un sistema in quanto contiene oggetti (siano essi tangibili o meno) definiti da attributi, che tramite un comparto di regole interagiscono tra di loro e con l'ambiente esterno. Scegliendo di interagire con questo sistema i giocatori si mettono nelle condizioni di vivere l'esperienza di un conflitto fatto di scontri veri e propri, i quali si svolgono chiaramente sul piano ludico, e che per questo motivo si possono definire "sicuri". La sicurezza di un conflitto vissuto all'interno del cerchio magico non è affatto una componente da sottovalutare: basta solo immaginare che cosa succederebbe se ogni volta che, durante una partita ad un qualche gioco da tavolo, la pedina che mi rappresenta finisse sulla casella che simboleggia la prigione e anche a me toccasse di scontare realmente del tempo in una prigione vera e propria. Questo esempio, volutamente parossistico, ci fa capire come, venendo meno la sicurezza e in questo caso anche l'artificialità del conflitto, verrebbe meno anche la spinta dell'individuo a mettersi in gioco e scontrarsi con altri individui. Più avanti sarà invece esplorata la modellazione di un gioco-artefatto-sistema, su di un piano matematico utile per lo studio vero e proprio di alcune delle proprietà che caratterizzano entità di questo tipo.

2.4 Game studies e Game Design

La disciplina che prende il nome di *Game Studies* può essere vista come lo sviluppo scientifico dell'interesse che sul finire del secolo scorso è andato concentrandosi intorno al gioco inteso sia come attività che come artefatto, e che ha ricevuto negli anni contributi significativi da sempre più numerose branche del sapere come la sociologia, la matematica, la pedagogia, la storia e via dicendo. L'evoluzione della civiltà moderna, per come l'abbiamo vissuta fino ad oggi, ha un ruolo fondamentale nella messa in evidenza del gioco come componente imprescindibile dell'essere umano. Via via nel corso degli ultimi decenni sono andati moltiplicandosi gli studi che mettono al centro della loro indagine il gioco, eplorando le relazioni che l'uomo intraprende con se, con gli altri e con l'ambiente mediante l'attività ludica, andando ad analizzarne le caratteristiche formali e perfino a ricostruire la storia di alcuni dei giochi più antichi e di come questi abbiano influito sui costumi umani. La civiltà ha un ruolo in questo processo in quanto fino a non molto tempo fa ha teso a separare il gioco dal resto, ha reso più evidente il confine da ciò che sta dentro il cerchio magico e ciò che ne sta al di fuori. Soltanto recentemente si è notata una felice apertura di quella parte di "universo" cosidetto *serio*, nei confronti del mondo ludico: un'apertura in parte dovuta al successo di alcuni giochi (e videogiochi in particolare) che sono riusciti a coinvolgere un numero enorme di persone (basti pensare ai milioni di utenti che giocano in rete a giochi come *World of Warcraft* [9] o simili), in parte probabilmente ad un'attenzione crescente riguardo alle tematiche dell'infanzia e dello sviluppo infantile e delle forti implicazioni che esistono tra il gioco e questi. Inoltre, sicuramente dovuta anche a opere che molto prima che tutto ciò avvenisse, hanno saputo sviluppare idee convincenti offrendo analisi anche molto approfondite e che hanno portato alla luce la continuità latente tra cultura e gioco. Opere come quella di Johan Huizinga, come il già citato "L'uomo e i giochi" di Caillois o ancora il lavoro svolto da John Nash sulla Game Theory e in particolare sui modelli di giochi non cooperativi [10]. Nash infatti dimostra come in ogni

interazione per così dire strategica, nella quale i decisori coinvolti siano a conoscenza delle regole che gestiscono tale interazione e delle conseguenze di ogni scelta possibile, esista almeno un punto chiamato "punto di equilibrio". Questo, che prende il nome di "equilibrio di Nash", rappresenta la situazione per cui ciascun individuo, adottando la strategia del massimo[3], ottiene il miglior risultato individuale possibile e al contempo il miglior risultato collettivo. Una situazione del genere può essere immaginata in chiave ludica: ovvero si può pensare di modellare l'interazione tra individui come un gioco (competitivo) nel quale i giocatori sono chiamati a intraprendere scelte individuali che hanno comunque un interesse collettivo. Un equilibrio si avrà quindi se, date le strategie dei vari giocatori, nessuno dei partecipanti può migliorare il proprio risultato cambiando soltanto la propria strategia. Una tale situazione costituisce di fatto la possibile soluzione di un gioco. L'importanza che il lavoro svolto da Nash ricopre nell'ambito dei game studies è data principalmente dal fatto che un tale livello di astrazione rende possibile l'applicazione del concetto di "punto di equilibrio" a qualsiasi tipo di gioco non cooperativo al quale possano giocare *n* giocatori, oltre a fornire un utilissimo approccio allo studio delle strategie adottabili.

Con Game Studies si tende quindi a indicare un insieme di contributi scientifici provenienti da diversi ambiti del sapere volti allo studio del gioco su più livelli (storico, formale, sociale) e che hanno come scopo ultimo l'indagine di tutti quei fenomeni che in qualche misura rientrano a far parte dell'attività ludica spontanea e non, esperita tramite un qualsiasi artefatto ludico. Strettamente legato ai Game Studies è il *Game Design*. I giochi, intesi come artefatti, sono in grado ormai di proporre al giocatore conflitti sempre più articolati e complessi e questo perché sempre più persone stanno dedicando una grande cura alla loro progettazione, prendendo spunto e partendo dagli studi sui giochi più significativi, traendo conoscenza dalle teorie più innovative sul gioco e sulle sue varie forme. La fase di progettazione, il design appunto, riveste o dovrebbe rivestire un ruolo di

3 Con "strategia del massimo" si intende l'insieme di scelte che porta un giocatore ad ottenere il massimo del risultato individuale ottenibile.

prim'ordine nella creazione per esempio di nuovi giochi da tavolo, e in maniera costante e sempre più attenta chi in questi anni si è occupato proprio di questo è andato a indagare un concetto basilare: l'esperienza. "Grazie alla conoscenza del Game Design come disciplina, dei suoi principi e dei suoi metodi, il game designer può creare giochi *belli* e *validi* in quanto tali, ovvero in grado di generare esperienze di gioco piacevoli e significative" [2]. Ecco perché sembra davvero utile che chi decide di sviluppare giochi si "nutra" delle basi teoriche gettate in precedenza, affinché chi gioca abbia la possibilità di esperire qualcosa che gli trasmetta un significato ("significative") e che lo diverta, affinché l'esperienza ludica sia in qualche modo attiva e costruttiva.

3 TIPOLOGIE DI GIOCO E LORO PROPRIETÁ

3.1 Giochi da tavolo: German vs American, una possibile classificazione

"The question most frequently asked by journalists and by new designers is about the original inspiration for the game—of the tiny spark of creation of the small universe that is the game. Which comes first? Is it the literary aspect (the theme), or the technical key (mechanics), that starts the machine which ends up forming the game?"[11]

Le considerazioni contenute nei paragrafi seguenti si riferiscono più in particolare ai giochi da tavolo, ma sono pur sempre applicabili ad altre tipologie di giochi come per esempio i videogiochi (diretti discendenti dei giochi da tavolo) o giochi in cui non è previsto un supporto materiale. Provando a inquadrare i giochi da tavolo senza abbandonare la felice classificazione del Caillois, senza dubbio essi ricadono a pieno titolo nell'ambito del ludus, essendo attività svolte volontariamente al fine di cimentarsi con un ostacolo appositamente creato. Un'attività ludica, quindi, che si piazza a mio avviso nei quadranti dell'agon e dell'alea, configurandosi di volta in volta in infiniti possibili equilibri di sorta, facendo prevalere una spinta o l'altra, mescolando in modo calibrato (nel caso di giochi ben progettati) la competizione e lo scontro diretto tra avversari e la sottomissione al giudizio imprevedibile e cieco della sorte. Chi gioca dunque ai giochi da tavolo dovrà, a seconda del tipo di gioco, essere abile nel prevedere le mosse altrui o nello scegliere le proprie in modo da ricavarne un profitto il più alto possibile, oppure dovrà essere fortunato nel lanciare un dado o nell'estrarre una carta favorevole da un mazzo di carte raffiguranti eventi di un qualche tipo, o ancora, come accennato sopra, una combinazione delle due.

Bruno Faidutti, designer di giochi affermato a livello internazionale, prosegue nel post citato in apertura del capitolo rispondendo alla domanda che a suo avviso è posta frequentemente dai giornalisti, ovvero se nella fase di creazione di un gioco siano i temi (le ambientazioni) o le meccaniche (il motore) a svolgere un ruolo di primo piano. La risposta che ci da Faidutti mette in luce una delle possibili classificazioni dei giochi da tavolo, più precisamente quella che vedrebbe l'insieme suddiviso in due macro sezioni: i giochi da tavolo stile *german*[4], e i giochi in stile *american*.

Anche se questa classificazione può apparire in qualche modo semplificativa, rispecchia la realtà contrapponendo due tendenze opposte e realmente esistenti. Due sottoinsiemi che, come succede per l'alea e per l'agon, sono concettualmente separati e allo stesso tempo complementari. Due poli, ancora una volta, di uno stesso sistema basato su possibili equibri, su vari gradi di mescolanza tra un'istanza (o meglio l'insieme di spinte che la compongono) e la sua contraria e complementare. Lo spettro di possibili combinazioni andrà da giochi a due giocatori che non presentano nessuna fonte di *randomness* (come dadi o mazzi di carte) e a informazione perfetta (ovvero i giocatori sono entrambi al corrente di tutte le scelte dell'avversario) tipo gli scacchi, a giochi come *Scale e serpenti* o il famoso *Gioco dell'oca,* in cui il controllo del giocatore è ridotto a zero e che vengono per questo definiti anche come non-giochi, dal momento che chi vi gioca in pratica assiste più che interagire attivamente con il gioco. In genere si dice che questi giochi *giocano* il giocatore, e non viceversa. Secondo il designer, la scuola cosidetta tedesca tende a produrre giochi basati sulle meccaniche, su insiemi di "ingranaggi" anche molto astratti che si configurano in sistemi fortemente caratterizzati da una componente matematica ben studiata. In questo genere di artefatti il tema finisce con l'essere un qualcosa di quasi decorativo, che ha lo scopo di accompagnare l'esperienza ludica ma che non ne costituisce il fulcro, il quale è dato appunto dai meccanismi di gioco e dall'insieme delle strategie

4 Anche conosciuti come *Euro-games*.

adottate dai giocatori. Dalla parte opposta si piazzano quindi i giochi pensati in stile americano, in cui si riscontra spesso un'attenzione piuttosto alta nei confronti del tema guida del gioco, e meccaniche che cercano di simulare il più fedelmente possibile le azioni legate al tema stesso ma che inevitabilmente rischiano di forzare lo svolgimento del gioco. In altre parole, se per i giochi in stile german vengono certamente prima le meccaniche di gioco, per i giochi in stile american l'importanza maggiore sembra avercela l'ambientazione dell'esperienza ludica, l'immaginario mondo di gioco che si viene a creare durante la partita. Lewis Pulsipher, in un suo post del 2006 [12] tenta di andare un po' più nel dettaglio e stila un elenco di proprietà che, secondo lui, dovrebbero costituire una sorta di decalogo del gioco in stile german. Tra le altre, alcune delle caratteristiche che secondo Pulsipher un gioco german presenta con più frequenza sono:

- Regole semplici;
- Un numero ragionevolmente basso di scelte possibili ogni turno;
- Impossibilità dei giocatori di essere eliminati dal gioco;
- Bassa interazione, viene evitato il conflitto aperto;
- Il tema vi appare come aggiunto, "superfluo";
- Tendenza a non usare molto i dadi;

In particolare è interessante sottolineare come la bassa interazione tra i giocatori, tipica di un gran numero di giochi stile german, renda questi stessi giochi più simili a una gara o a un solitario multi-giocatore e, allo stesso tempo, uno scarso utilizzo di fonti di randomness come i dadi garantisca la necessità dei giocatori di dover adottare una tattica individuale per riuscire a vincere. Alcuni esempi di titoli german di grande successo: *Carcassonne* [13], un classico tra i giochi di piazzamento tessere in cui durante la partita i giocatori costruiscono turno dopo turno il terreno sul quale stanno giocando, posizionando appunto le tessere che raffigurano il paesaggio ispirato alla famosa cittadina fortificata del sud della Francia dal quale il gioco stesso prende il nome. Un altro titolo di grande successo e del quale sono state pubblicate negli anni varie estensioni è *I coloni di Catan* [14] (titolo originale *The Settlers of Catan*), o ancora *Caylus* [15] nel quale i giocatori sono chiamati a gestire la

forza lavoro impiegata nella costruzione di un nuovo castello per Re Filippo IV il bello, cercando di fare buoni affari e di accaparrarsi i favori dello stesso Filippo.

In giochi appartenenti alla scuola americana queste caratteristiche, come si è detto sopra, non si riscontrano se non in minima parte. Sono infatti giochi che, oltre all'alta aderenza delle meccaniche al tema, contengono mediamente una più alta interazione diretta in cui i giocatori sono portati a scontrarsi o allearsi tra di loro mentre si immergono nel brivido dell'alea. Un esempio di gioco che rispecchia questo stile è il famosissimo *Risiko* [4], nel quale ogni battaglia è decisa dai dadi, oppure *Hotels* [16], che vede i giocatori impersonare magnati vogliosi di costruire alberghi di lusso per poi farci affari: in questo gioco il risultato del lancio dei dadi determina marcatamente l'economia di risorse a disposizione di ogni giocatore, il quale per cercare di superare in profitti gli avversari deve necessariamente rischiare in prima persona.

3.2 Un altro livello di classificazione: competizione vs coordinamento

Aldilà delle caratteristiche che fanno rientrare un gioco in uno stile invece che in un altro, esiste un'altra dicotomia che rende possibile un'ulteriore classificazione: in particolare è possibile suddividere l'universo ludico in giochi di competizione e giochi di coordinamento. Da questo punto di vista è allora interessante andare a vedere se a guidare le scelte dei giocatori è appunto un obiettivo individuale, più o meno in contrasto con gli obiettivi degli altri partecipanti, o se l'obiettivo è invece uno solo e condiviso dalla totalità dei giocatori. Ancora una volta le due direttrici non si escludono a vicenda, ed è possibile che a seconda dell'equilibrio che si viene a creare un gioco sarà ascrivibile a una categoria o all'altra. Alcuni esempi possono chiarire il concetto: un tipico gioco di competizione è *Morra Cinese* (altrimenti conosciuto come *Sasso-*

Carta-Forbice), in cui anche se è possibile pareggiare, l'esito che vede un giocatore vincente non può certo rendere soddisfatto l'avversario. Contrariamente, in un gioco come *Caccia al cervo* [2][5] esiste un chiaro obiettivo di gruppo, ovvero giocare tutti cervo in modo tale da accordarsi sulla caccia da fare e guadagnare tutti lo stesso punteggio; ma al contempo esiste un altro obiettivo che può spingere i giocatori a fare una scelta di tipo assicurativo e a giocare *lepre* per garantirsi di ricevere comunque un pagamento, anche se inferiore a quello ottenibile se tutti quanti si accordassero su *cervo*. In generale, come si accennava poco sopra, sono solitamente compresenti soprattutto nei giochi di coordinamento sia la spinta alla cooperazione tra giocatori, sia un certo incentivo a un'azione che porti vantaggi personali. Più il gioco tende a incentivare una spinta e, di conseguenza, a frenare la spinta opposta, e più si caratterizzerà per essere maggiormente un gioco di competizione o un gioco di coordinamento. Infine una precisazione: quando ci si riferisce alla competizione non si intende per forza un conflitto diretto, ma si fa riferimento invece a ogni tipo di situazione dove due o più istanze si fronteggiano per accaparrarsi il risultato più vantaggioso e lasciare il più possibile a bocca asciutta l'avversario.

È possibile stilare un sintetico elenco dei vari sotto-tipi di giochi competitivi e di coordinamento. Generalmente i giochi di conflitto possono vedere i giocatori impegnati a scontrarsi tra di loro confrontandosi direttamente, oppure competere indirettamente in una sorta di gara, questo, come si è detto a seconda del grado di interazione presente nelle meccaniche di gioco. Per quanto riguarda invece i giochi di coordinamento, anche detti giochi cooperativi (o in inglese *co-op boardgames*), i giocatori possono essere chiamati ad agire nell'ambito di un'unica squadra intenta a sconfiggere il gioco stesso, oppure a dover sconfiggere la squadra avversaria; alcuni di questi giochi prevedono anche che ci siano uno o più traditori i quali segretamente agiscono contro tutti gli altri giocatori fungendo da antagonisti insieme al gioco. In questo caso le dinamiche della partita vedranno la

5 In questo volume si fa un'interessante rassegna di giochi classici presentandoli in maniera schematica e sottolineandone le varie caratteristiche.

squadra composta da tutti i giocatori impegnata a superare le difficoltà imposte dal gioco e allo stesso tempo a dover intercettare le mosse di uno o più giocatori avversari, in una sorta di compromesso tra la prima e la seconda sotto-categoria.

3.3 Che cosa rende un gioco divertente: le proprietà che un gioco dovrebbe avere per essere un buon gioco

Scegliere di giocare a un gioco è senz'altro una scelta che può essere guidata dai più diversi motivi. Sicuramente però, chi la compie si aspetta di ricavare qualcosa da questa attività. Si può giocare per distrarsi, per rilassarsi, per il gusto di competere con sé o con gli altri oppure, più generalmente, per divertirsi. L'esperienza ludica, se fatta tramite un artefatto ben progettato, può e anzi dovrebbe veicolare qualcosa di significativo. Affinché un gioco possa essere in grado di offrire un'esperienza del genere dovrà possedere alcune proprietà invece di altre. Le proprietà di cui si parla in questo capitolo si riferiscono in generale agli artefatti ludici, e più precisamente al *game-play* che ne può scaturire, all'esperienza che ne può risultare quando qualcuno vi gioca. Un buon gioco è sostanzialmente un gioco a cui si gioca volentieri: soprattutto un buon gioco dovrebbe essere in grado di offrire un'esperienza significativa a chi decide di giocarvi e che pertanto, sarà portato a scegliere di giocarvi di nuovo. A questo proposito si parla di grado di rigiocabilità che un artefatto ludico è in greado di esprimere. In questo senso quindi, tra le proprietà che un gioco può modulare con le sue proprie caratteristiche ve ne sono alcune sulle quali è opportuno spendere alcune parole[6]:

6 Per uno spunto sulle proprietà che rendono un gioco rigiocabile si rimanda al sito Ministry Of Board Game disponibile a: http://www.mofbg.co.uk/2014/09/lessons-on-re-playability.html [consultato il 31 gennaio 2015]

1. durata del gioco e delle sue fasi;
2. varietà (del gioco);
3. randomizzazione della partita e delle sue fasi;
4. differenti strategie vincenti;
5. abilità di gioco cooperative o competitive;

1) **Durata della partita** è fondamentale ai fini di garantire un'esperienza ludica ottimale. Il tempo speso a giocare deve sempre essere preso in considerazione, deve essere chiaramente quantificabile e proporzionale alla complessità del gioco. Esistono giochi che prevedono un tempo prestabilito per le partite, spesso scandito da appositi oggetti come timer o clessidre, e ci sono giochi in cui invece il tempo del singolo turno o della partita non è predefinito e i giocatori non hanno in questo senso nessun limite. Per entrambe le tipologie di gioco è comunque imoportante che il tempo impiegato (imposto) dai (ai) giocatori per portare a termine una partita sia ragionevole. Al tempo di gioco sono legate una serie di problematiche di notevole importanza: basta immaginarsi un gioco che, se giocato in un certo numero di persone per esempio, si protragga per più giorni indefinitamente. Da tenere in grande considerazione anche il tempo che intercorre tra la designazione del vincitore e la fine effettiva della partita: un tempo ottimale in questo caso sarebbe un tempo nullo, che faccia quindi andare a coincidere i due momenti. Questo perché sarebbe indubbiamente deleterio se si conoscesse il futuro vincitore per esempio dopo soltanto pochi turni dall'inizio di una partita. C'è inoltre il problema, di estrema rilevanza, legato all'eliminazione di uno o più giocatori dalla partita: immaginiamoci allora cosa potrebbe significare venire eliminati durante i primi turni di una partita che dura un tempo molto lungo, la nostra esperienza in quel caso non potrà certo essere positiva. A volte il tempo è un elemento che viene sfruttato per il funzionamento di una meccanica. Per esempio ai giocatori può essere richiesto di eseguire delle scelte o delle azioni contemporaneamente e in un determinato tempo prefissato, e questo può servire sia per ottenere l'incompletezza di informazioni a disposizione di ogni singolo giocatore in una data fase di gioco, sia per scandire il

ritmo di una partita (ci sono giochi che sono studiati per durare dieci turni, come giochi che invece sono pensati per durare ad esempio esattamente un'ora). Generalmente, rifacendosi alla classificazione proposta in precedenza, i giochi in stile german si caratterizzano per una durata media intorno a un'ora, che può essere considerata una durata relativamente breve. Questo stile di progettazione infatti predispone solitamente delle meccaniche che non permettono alle partite di protrarsi troppo nel tempo, come per esempio succede in *Carcassonne* con le tessere che i giocatori sono chiamati a piazzare: una volta posizionata l'ultima infatti il gioco termina e si procede al conteggio finale dei punti. Questa proprietà (come le altre che vedremo più avanti) costituisce un notevole fattore di rigiocabilità: in fase di design infatti, riuscire a calibrare in modo ragionevole quello che sarà il tempo stimato necessario per portare a termine una partita può dare molte chance a un gioco di offrire un'esperienza significativa e, pertanto, di incentivare i giocatori a scegliere di giocarci nuovamente.

2) **Varietà del gioco**: si intende qui con varietà del gioco la qualità grazie alla quale lo stesso artefatto si rivela in grado di offrire esperienze sensibilmente diverse a seconda dell'interpretazione che ne viene data, o a seconda della versione differente alla quale si gioca di volta in volta. In altre parole un gioco esprime la sua varietà quando al suo interno sono contenuti scenari e ambientazioni alternativi, oppure set di regole alternative che senza stravolgere la natura del gioco stesso modificano in modo sensibile le dinamiche di gioco (creano una variante del gioco stesso). Un esempio può essere rappresentato da un mazzo di carte da gioco le quali, a seconda delle regole che si intende seguire, offriranno una gamma decisamente vasta di possibili giochi diversi tra loro. Lo stesso si può dire per giochi che non prevedono oggetti materiali per essere giocati, per i quali risulta forse anche più semplice dal momento che spesso questi ultimi si basano su un numero molto ristretto di meccaniche sinteticamente racchiuse in poche regole semplici da memorizzare: allora possiamo scegliere se giocare a *Guardie e Ladri* dove sono due squadre a fronteggiarsi,

oppure al classico *Nascondino* dove invece si ha un solo giocatore impegnato nello scovare i nascondigli degli avversari in una sorta di uno contro tutti modulato sulla stessa meccanica. Ad ogni modo la varietà di un gioco è ciò che lo rende adattabile alle esigenze di chi ci vuole giocare, e che quindi ne determina fortemente sia la giocabilità che, forse in modo ancora più marcato, la ri-giocabilità. In linea generale un'architettura di gioco troppo rigida difficilmente permetterà al gioco stesso di esprimere un alto grado di varietà al suo interno. In fase di progettazione sarà quindi utile per il designer cercare di mettere insieme un comparto di regole e di eventuali materiali necessari per la fruizione del gioco stesso che possa garantire un certo grado di libertà all'interno del game-play. Si possono poi per esempio prevedere parti aggiuntive, regole diverse e così via, come se a un unico gioco ne corrispondessero più di uno. Un po' come se nella stessa scatola, reale come nel caso di un gioco da tavolo, o immaginaria se ci riferiamo ad altri tipi di giochi, fossero contenuti più giochi più o meno simili tra loro. La varietà di gioco non è quindi semplice da ottenere, ma allo stesso tempo costituisce un'ottima qualità da possedere.

3) **Randomizzazione del gioco e delle sue fasi:** proprietà che il gioco possiede quando durante una singola partita gli obiettivi, le risorse a disposizione dei giocatori o gli eventi che scaturiscono dal gioco stesso (come per esempio gli imprevisti del *Monopoly*) sono assegnati in maniera casuale. Si parla di questa proprietà riferendosi anche alle singole fasi del gioco in quanto ognuna di queste può contenere o meno una o più fonti di randomness. Tipiche fonti di eventi casuali sono i dadi che i giocatori lanciano ad esempio per determinare le sorti di una battaglia (come in *Risiko*), un mazzo di carte dal quale i giocatori devono pescare per ricevere il proprio *bonus* (o *malus* nel caso il gioco lo preveda), o ancora un set di regole astratte che a seconda dello stato attuale del gioco[7] assegni a uno o più giocatori una particolare risorsa o abilità nel caso per esempio dei giochi di ruolo. Generalmente tali fonti di randomness non possono coesistere se non in maniera ben progettata, in

7 Più avanti si riprenderà questo concetto spiegando più in dettaglio che cosa si intende con stato attuale del gioco. Vedi cap 3.

modo cioè da lasciare comunque spazio all'azione del giocatore. Alcuni giochi in stile american, specialmente quelli che meglio rappresentano questo stile, lasciano poco se non pochissimo spazio decisionale al giocatore che per questo si trasforma in una sorta di "giocatore/spettartore". Possedere una fonte di casualità non è certo indispensabile per un gioco: basti pensare ai molti giochi, anche classici, che non ne prevedono affatto come il già citato gioco degli Scacchi. Al contrario, contenere una o più di queste fonti significa che in una o più fasi della partita il ventaglio delle azioni o degli eventi possibili sarà aumentato o diminuito, a seconda dell'utilizzo che ne viene fatto: esistono infatti scelte di design tali che, per esempio, sfruttando il lancio di un dado riescono in qualche modo a restringere il campo di azione di un singolo giocatore. Queste scelte sfruttano, a mio avviso in modo innovativo, tali fonti di casualità impiegandole per esempio in maniera da imporre un certo ritmo al singolo turno durante il quale i giocatori sono chiamati a fare delle scelte. Non solo, se lanciare molte volte un dado in qualche modo tende ad ampliare lo spazio di possibilità aumentando il numero di possibili cammini che portano a uno stesso risultato, sfruttando lo stesso lancio di dadi per esempio all'interno di una meccanica che gestisce le possibili scelte di un giocatore si può riuscire a limitare il numero delle scelte stesse. Un meccanismo come quello del *number-matching* può servire come esempio: in alcuni giochi [17], per limitare la scelta tra un certo numero di possibili azioni, si può pensare di etichettare queste stesse azioni e, una volta lanciato il dado, imporre al giocatore l'azione che presenta l'etichetta corrispondente al risultato del lancio. In questo modo, di fatto, una fonte di casualità come il dado riesce a modulare lo spazio di possibilità in un ottica di riduzione. In un suo articolo il designer Christian Strain [18] sostiene che esistano principalmente quattro macro aree dove è possibile inserire una fonte di casualità:

- Durante il setup[8];
- Prima che una decisione venga presa;
- Dopo che una decisione venga presa;
- Alla fine della partita;

Un esempio di randomizzazione durante il setup può essere rappresentato dalla scelta casuale dell'ordine di turnazione; molto diffusa anche, per esempio, la distribuzione a ciascun gicatore di un numero prefissato di carte prese casualmente da un mazzo. Due scelte opposte invece quelle che vedono il piazzamento della fonte di casualità prima o dopo una decisione del giocatore: non è certo la stessa cosa immaginare la propria strategia prima o dopo di aver saputo se avremo a disposizione un certo numero di risorse o se invece non ne avremo affatto. In entrambi i casi la randomizzazione andrà senz'altro a influire sull'esito della partita, e in particolare del turno in questione, ma allo stesso tempo l'impatto sulle scelte tattiche che via via i giocatori sono portati a fare sarà drasticamente differente. Infine è possibile scegliere di inserire un elemento che generi randomness alla fine di un gioco: questa scelta presenta però alcune criticità ben evidenziate da Strain nel suo articolo. Così scegliendo, infatti, si corre il rischio di sottrarre importanza a tutti gli sforzi fatti dai giocatori per arrivare, per esempio, al combattimento finale. Affidare le sorti di una partita ad un evento casuale rischia di compromettere a fondo la giocabilità di un gioco; si rende opportuno quindi cercare di mitigare il potenziale di questa fonte di casualità offrendo ai giocatori qualche meccanismo tramite il quale prepararsi ad un tale evento, anticiparne qualche aspetto o addirittura di eliminarlo del tutto. Se invece ci si riferisce al singolo turno e lo si osserva come una sequenza di eventi e scelte correlate l'una all'altra, si può notare come questa catena sia fondamentalmente composta da istanze casuali, possibilmente casuali oppure da scelte volontarie. In un esempio: durante il mio turno mi viene chiesto di lanciare un dado (elemento *random*), a seconda del risultato potrò scegliere se lanciare nuovamente (possibile *random*) oppure se attivare una certa carta

8 Con *setup* si intendono tutte le azioni da compiere affinché il gioco sia configurato in uno dei possibili stati iniziali.Nel caso di Risiko per esempio il setup prevede la distribuzione di un certo numero di carte e armate a ciascun giocatore, il piazzamento di queste sul tabellone di gioco e infine di stabilire l'ordine di turnazione dei giocatori.

all'interno di una rosa di *n* carte disponibili (scelta volontaria). A seconda dell'ordine in cui queste istanze si manifestano al giocatore di turno, l'esperienza di gioco può mutare radicalmente dando più o meno la sensazione al giocatore di avere in mano la propria partita [18]. In tutti i casi di studio che saranno presentati nell'ambito di questo elaborato sono state fatte scelte di design che vanno a inserire fonti di randomness in una di queste quattro macro aree di gioco. Alcune delle scelte fatte sono servite per tentare di simulare o forzare un comportamento verosimilmente umano, mentre altre sono state fatte perché ritenute buone al fine di riprodurre modelli di gioco il più possibile giocabili. Altre ancora invece sono state il vero oggetto del caso di studio, come per esempio nei test fatti sul modello del *Gioco dell'Oca*.

4) **Differenti strategie vincenti**: un gioco dovrebbe offrire al giocatore la possibilità di scegliere tra tattiche differenti che in modo più o meno agevole lo possano portare a ottenere un certo risultato. Si può pensare per esempio di affidare ai giocatori diversi obiettivi da raggiungere senza però tralasciare il fatto che la diversificazione dei *goals* non dovrebbe interferire sul livello di interazione generale contenuto nel gioco. In Risiko ognuno deve perseguire un obiettivo che, per quanto possa essere diverso da quello degli avversari, non fa venire meno l'elemento del conflitto aperto tra i giocatori, ma anzi può in alcuni casi addirittura incentivarlo. L'effetto di una tale scelta di design si riflette inevitabilmente sulle strategie dei giocatori in quanto una determinata serie di mosse efficaci per un dato obiettivo, può naturlamente non esserlo altrettanto per un altro. Aumentano quindi le possibili strategie vincenti per ogni singolo giocatore, supponendo che questi riceva ad ogni partita un goal diverso, le informazioni raccolte alla partita giocata in precedenza non saranno fruibili o lo saranno solo in parte. In un certo senso questa proprietà è fortemente legata al grado di rigiocabilità di un gioco: la difficoltà di trovare una strategia vincente può rivelarsi un buon incentivo a giocare nuovamente nella speranza di padroneggiare sempre di più le dinamiche di gioco, le regole, e essere quindi in grado di compiere scelte più efficaci. Comprendere a fondo un gioco e le sue meccaniche

significa poter pianificare un set di possibili mosse in modo più efficiente, per questo un gioco per il quale la curva di apprendimento[9] risulta essere bilanciata esprimerà un grado di rigiocabilità più alto rispetto a un gioco nel quale, per esempio, in un numero molto ristretto di partite o addirittura di turni si riesca a raggiungere il goal con facilità. Alcuni giochi, specialmente alcuni videogiochi, sono spesso progettati in modo da mettere il giocatore di fronte a sfide via via più complesse: gli obiettivi iniziali servono spesso per imparare alcune dinamiche basilari che poi il giocatore dovrà riuscire a mettere in pratica negli step successivi per i quali i goals risulteranno chiaramente più difficili da raggiungere [19]. Questa struttura a "gradini" permette di rallentare il processo di apprendimento, nella fattispece "spezzettandolo" in vari sotto-obiettivi via via sempre più complessi i quali, in sostanza, spingono il giocatore a cercare sempre nuove tattiche e probabilmente ad ottenere un'esperienza di gioco tendenzialmente meno ripetitiva.

5) **Abilità di gioco cooperative/competitive**: più sopra si è parlato della differenza che esiste tra giochi di coordinamento e giochi di competizione e di come queste tipologie pongano l'enfasi su un aspetto o su un altro. Mentre un gioco di competizione deve offrire un set di meccaniche che gestiscano in modo corretto gli scontri tra giocatori, un gioco cosiddétto *co-op* deve necessariamente essere progettato sfruttando meccaniche che mettano i giocatori in condizione di poter scegliere se collaborare con il proprio compagno di squadra (in un gioco che prevede la suddivisione dei giocatori in squadre) oppure di non collaborare ad esempio con il giocatore che si ritiene essere un *traditore* (nei giochi che prevedono questo tipo di avversari). Esistono varie problematiche legate ai giochi cooperativi e una di queste è senz'altro l'uscita precoce dei giocatori dalla partita. L'eliminazione dei giocatori è un problema che sussiste anche in giochi competitivi, ma se in questi ultimi lo scopo è generalmente modulato proprio sul "battere" l'avversario possibilmente mettendolo fuori gioco, in giochi che invece dovrebbero spingere i partecipanti a

9 Con curva di apprendimento si intende qui il rapporto tra tempo necessario all'apprendimento e mole di informazioni correttamente interiorizzate.

collaborare sarebbe preferibile non perdere preziosi compagni di squadra lungo il tragitto. Sia l'esperienza del giocatore che viene eliminato, sia quella dei giocatori che rimangono in gioco, verranno influenzate in maniera tendenzialmente negativa da un simile evento; per questo alcuni designer scelgono sempre più spesso di non permettere l'eliminazione dei giocatori (questo succede specialmente nei giochi in stile german) o tentano di posticiparla il più possibile adottando diverse strategie di implementazione (questo aspetto sarà oggetto di discussione nel terzo caso di studio). Altri due problemi legati a entrambe le tipologie di gioco sono il cosiddetto *runaway leader* e il *leader bashing*: queste due dinamiche di gioco possono essere considerate una la causa dell'altra. *Runaway leader* designa la situazione in cui un giocatore riesce a creare un divario incolmabile tra sé e gli avversari; nel caso in cui invece il distacco fosse in qualche modo recuperabile, i giocatori che sono in svantaggio si potrebbero trovare a dover compiere azioni che rallentino la "fuga" del leader per garantirsi la possibilità di vincere, e questo è tipicamente chiamato *leader bashing*, o "attacco al capo". In questo caso il gioco demanda ai giocatori il compito di bilanciare la partita.

4 Modellazione e model checking

4.1 Modelli e sistemi transizionali

Con questo capitolo si intende chiarire alcuni dei concetti alla base del lavoro di sperimentazione svolto, nonché mettere in luce alcuni aspetti delle metodologie utilizzate al fine di fornire un quadro generale utile a una migliore comprensione. Tramite l'utilizzo di strumenti di model checking [19] sia probabilistico che statistico e la modellazione di alcuni giochi si è inteso studiare alcune delle proprietà di cui al capitolo precedente, con l'obiettivo di individuare una metodologia adatta a una progettazione efficace di artefatti ludici. Per prima cosa è utile rendere esplicito ciò che qui si intende con modello di gioco. Un modello di gioco corrisponde a un sistema di transizioni: ossia è descritto da un insieme di variabili che rappresentano i vari elementi del gioco (posizione dei giocatori sul tabellone di gioco, pedine disponibili e così via) e da un insieme di transizioni che rappresentano i cambiamenti dei valori di tali variabili durante il gioco. Uno stato del sistema di transizioni è quindi una combinazione di valori assegnati alle variabili del sistema stesso. Provando a essere più formali, dato un insieme di variabili X, un modello è definito da:

- un insieme finito di stati S_x che descrive tutti i possibili assegnamenti delle variabili;
- un insieme finito di transizioni che descrivono il passaggio da uno stato all'altro;

Per realizzare i vari modelli di gioco si è scelto di utilizzare PRISM, una piattaforma versatile che tramite l'utilizzo di alcune parole chiave rende possibile la descrizione, oltre che dei modelli stessi, delle proprietà che si intende far verificare al *model-checker*. I modelli descritti sono di tipo probabilistico, ovvero a ogni configurazione (stato), possono corrispondere una o più possibili transizioni le quali sono globalmente regolate da probabilità definite in fase di implementazione del modello; in PRISM è infatti possibile "guidare" le transizioni associandole a una data probabilità e,

nel caso in cui sia prevista una sola possibile transizione, è legittimo omettere la probabilità da associarvi la quale verrà impostata in automatico pari a uno. Le singole probabilità di ogni set di transizioni descritto devono sommare sempre a uno, in caso contrario in fase di costruzione del modello verrà segnalato un errore. In questo modo quindi, tramite la descrizione delle regole necessarie per il corretto svolgimento del gioco, è possibile creare un modello che riproduca i meccanismi alla base di un gioco da tavolo e, allo stesso tempo, studiarne le proprietà che ne guidano le dinamiche tramite l'esplorazione dei possibili cammini. Proprietà di tipo probabilistico, come "il sistema genera risultati positivi con probabilità P = *x*" oppure "con quale probabilità dopo la terza transizione la variabile *y* assume un valore maggiore di *n*?", dovranno essere descritte con proposizioni espresse tramite un opportuno linguaggio logico (logica temporale). Allo stesso modo dovranno essere descritte eventuali proprietà quantitative di interesse.

4.1 Model checking: ambiti di utilizzo e risultati raggiunti

Il model checking consiste nell'applicazione di metodi matematici formali al fine di verificare la correttezza di un dato sistema. La verifica di un sistema può essere vista sinteticamente come segue: dato un modello *M* di un sistema e una proprietà *p*, si intende verificare formalmente se *M* soddisfa *p*. Le verifiche vengono condotte su modelli di sistemi, come per esempio catene di Markov a tempo continuo o discreto, proprio perché come già accennato in questo modo si è in grado di rappresentare realisticamente una gamma molto vasta di sistemi [20]. Supponiamo di dover implementare un sistema in grado di gestire le partenze e gli atterraggi degli aerei di un aeroporto: cosa accadrebbe se in fase di testing venissero coinvolti direttamente i veivoli veri e propri? La risposta è ovvia: si rischierebbe un disastro. Un altro esempio che può sottolineare l'importanza della modellazione e della verifica formale delle proprietà di un sistema è la loro applicazione nell'edilizia: fare verifiche statiche sull'edificio in fase di progettazione può garantire che una volta

costruito per esempio il secondo piano, questo non vada a schiacciare il piano sottostante, o ancora che un solaio sia in grado di sostenere un certo peso e così via. In modo analogo il model checking fa lo stesso genere di verifiche andando però ad analizzare le dinamiche di un sistema modellato. La modellazione e la verifica formale dei modelli si rende quindi necessaria al fine di evidenziare possibili criticità nel funzionamento dei sistemi che si intende realizzare. Se questo è vero quando ci si riferisce alla fase di progettazione di un sistema, lo è altrettanto se si pensa di utilizzare il model checking come strumento per risolvere problemi che nascono dalla reale applicazione di un dato sistema. Immaginiamo ora che un certo modello di bicicletta sia stato messo in commercio e che, per un errore di progettazione, queste si muovano a marcia indietro anziché normalmente in avanti. È possibile tentare di risolvere il problema creando un modello della bici in questione per il quale sia possibile verificare se possiede o meno determinate specifiche, per esempio quella di far girare le ruote in avanti in conseguenza della pedalata. Una volta espressi i requisiti che il sistema deve possedere in una logica temporale adeguata, è possibile in definitiva utilizzare uno strumento di model checking col fine di verificare formalmente se il dato sistema soddisfa o meno tali requisiti. Per questi motivi (e non solo) le tecniche di model checking e verifica formale sono entrati nel corso degli anni a far parte di sempre più numerose pratiche industriali e di ricerca, a partire dalla produzione di nuovi componenti hardware (come chip e processori) o software per finire alla creazione di nuove molecole utili per esempio nella ricerca farmacologica. Una progettazione per così dire "sicura", o un metodo per la risoluzione di problemi anche molto complessi, sono senza dubbio un incentivo notevole ad affrontare sfide sempre nuove: nell'ambito di questo elaborato si è scelto di utilizzare alcune di queste tecniche per rendere più efficace la progettazione di artefatti ludici o per studiarne eventuali criticità al fine di offrire a chi vi volesse giocare un'esperienza la più positiva possibile.

4.2 Model checking probabilistico e model checking statistico

Non è sufficente a volte sapere se un dato sistema soddisfa o meno certi requisiti poiché può rendersi necessario sapere nel dettaglio in quanti casi ciò avviene e in quanti no. Al fine di rendere quantificabili i risultati di una verifica formale è possibile affidarsi a tecniche di model checking probabilistico o statistico. Le prime differiscono dalle seconde per il fatto che i risultati che restituiscono si basano sul totale effettivo delle possibili esecuzioni del sistema, anche dette cammini[10]. Se, come si vedrà in due casi di studio su tre, non è possibile effettuare una verifica probabilistica a causa per esempio dell'eccessiva grandezza del modello implementato, è comunque sempre possibile, tramite il metodo della simulazione, effettuare verifiche di tipo statistico: ovvero il model checker restituirà un risultato ottenuto verificando una data proprietà su un numero predefinito di cammini. Questo problema viene generalmente indicato come "exponential state explosion"[21], e si presenta quando la complessità del sistema implementato aumenta a dismisura la grandezza del modello impedendo al model checker di effettuare correttamente le verifiche richieste.

Una tipica problematica studiata attraverso la modellazione e il model checking è la *reachability* [22], cioè la possibilità che il sistema raggiunga un determinato stato. Più precisamente la "raggiungibilità" probabilistica indica la probabilità, partendo da uno stato S_x, di raggiungere un set di stati per i quali una data proprietà risulta essere vera. Un'esempio potrebbe essere: "quale è la probabilità che, partendo dallo stato iniziale, il sistema raggiunga uno stato S_i nel quale la variabile y ha valore $\geq n$?".

Il concetto di raggiungibilità consente di rappresentare due tipi di proprietà di fondamentale

10 Un *cammino* è l'insieme delle configurazioni (degli stati) e delle transizioni che descrivono una possibile esecuzione, che nel caso di un modello di gioco può essere vist come una possibile partita.

importanza al fine di stabilire la correttezza di un sistema: le *safety properties*, le quali per essere soddisfatte richiedono che durante l'esecuzione niente di negativo accada, ovvero che non si raggiunga nessuno stato corrispondente a una configurazione erronea, e le *liveness properties*, le quali richiedono invece che durante l'esecuzione prima o poi qualcosa di *buono* necessariamente accada. Per quanto riguarda il primo tipo di proprietà, un esempio particolare e insieme fondamentale per stabilire il corretto funzionamento di un sistema è l'impossibilità del sistema di raggiungere una qualche situazione di *deadlock*. Tali situazioni corrispondono a stati del sistema in cui non ci sono transizioni disponibili. Supponiamo di avere un modello di gioco che preveda un tabellone sul quale i giocatori siano chiamati a posizionare le proprie pedine: ciascun giocatore, prima di passare il turno al giocatore successivo, dovrà necessariamente piazzare una pedina sul tabellone. Non appena uno dei giocatori avrà terminato le proprie pedine il sistema resterà inevitabilmente in attesa che questi ne posizioni una nuova senza che ci sia però la possibilità che ciò avvenga: una situazione del genere bloccherebbe di fatto l'esecuzione. Per evitare che questo si verifichi è opportuno che il sistema sia in grado di gestire l'esaurimento delle pedine senza che l'esecuzione venga interrotta, magari non dando al giocatore la possibilità di giocare (saltando quindi il suo turno) oppure facendo terminare la partita.

Garantire che un modello soddisfi una proprietà del secondo tipo invece significa che durante l'esecuzione del modello si raggiunge, indipendentemente dalla lunghezza del cammino percorso, uno stato in cui qualcosa di positivo avviene: in altre parole un sistema che soddisfi tale proprietà è un sistema in grado di "progredire". Tornando all'esempio di prima, ogni posizionamento correttamente effettuato può essere considerato come evento positivo, pertanto un'esecuzione in cui nessun giocatore passa il turno al successivo senza prima aver piazzato la propria pedina soddisfa una prprietà di *liveness*.

Un ulteriore problema possibile da studiare tramite il model checking è la *resource starvation*.

Questo tipo di problema insorge quando un sistema che include componenti che agiscono in concorrenza tenta di distribuire un dato numero finito di risorse tra i componenti stessi senza riuscrirvi. Per un esempio che ne illustri i dettagli in modo chiaro si rimanda a [23].

4.3 Esempio di modello di gioco

Al fine di fornire un esempio di modello di gioco si propone la possibile implementazione di un modulo (Figura 4.1) che simuli un gioco nel quale due giocatori cercano di ottenere un risultato più alto dell'avversario lanciando ciascuno un dado a sei facce. Successivamente si mostrerà il grafo che rappresenta tutte le possibili esecuzioni e infine si verificherà se il modello soddisfa alcune proprietà. Le regole del gioco sono estremamente semplici: entrambi i giocatori possiedono un dado; il giocatore uno lancia il suo dado ("dado1") e così fa anche il giocatore due ("dado2"); se il risultato è lo stesso si gioca nuovamente fintanto che uno dei due non ottiene un numero più alto dell'avversario. Il modulo si apre con la dichiarazione del tipo di modello che si intende creare, in questo caso si tratta di un **dtmc** (ovvero un Discrete Time Markov Chain) [24]; subito dopo vengono dichiarate e inizializzate le variabili che rappresentano le varie istanze (giocatori, dadi e così via) che nell'esempio sono state considerate globali e quindi comuni a tutto il modello. Il modulo nominato **Partita** costituisce il motore vero e proprio del gioco e gestisce lo svolgersi delle sue diverse fasi. Al suo interno si trovano le transizioni che corrispondono ai possibili risultati dei due dadi e alla designazione del vincitore. In particolare le prime due transizioni hanno il compito di alternare il lancio dei due dadi, mentre le tre successive servono per assegnare la vittoria a uno dei due giocatori o, in caso di pareggio, per far tirare nuovamente i dadi. Ogni transizione è composta da una guardia che esprime una condizione sulla configurazione del sistema: se la guardia risulta vera allora il modello seleziona di volta in volta uno dei possibili assegnamenti con una

probabilità pari a quella indicata. Selezionando una delle transizioni disponibili il modello determina di fatto la sua configurazione successiva. Nel caso in cui la guardia risulti falsa il modello ignorerà l'intera transizione per passare a quella successiva. Nel caso dei dadi si può notare come ogni possibile risultato abbia probabilità 1/6, mentre per le transizioni relative alla gestione del vincitore si è omesso di definire una qualche probabilità che in questo modo risulterà pari a uno. Ad ogni modo le tre transizioni che servono a designare il vincitore si escludono a vicenda e coprono tutti i casi possibili, pertanto ogni risultato sarà "catturato" dal modello. In 4.2 si riporta il grafo relativo alle possibili esecuzioni del gioco mostrato in precedenza: i nodi del grafo rappresentano gli stati mentre le frecce rappresentano le transizioni. Ogni stato contiene una tripla $\{i, j, k\}$ che rappresenta i valori relativi nell'ordine alle variabili *dado1*, *dado2* e *vincitore*. Si sono omessi per questioni di spazio e leggibilità gli stati e le transizioni relativi ai valori di *dado1* compresi tra due e cinque. Nello stato iniziale le due variabili relative ai dadi e la variabile *vincitore* valgono zero. Con probabilità 1/6 (le probabilità legate alle transizioni sono scritte nei riquadri rettangolari) la variabile *dado1* assume un valore compreso tra 1 e 6. Lo stesso fa la variabile *dado2*. In caso di pareggio il sistema torna alla configurazione iniziale. Nel caso in cui invece uno dei giocatori ottiene un punteggio più alto la variabile *vincitore* prende valore pari a uno o a due, a seconda del giocatore che l'ha ottenuto.

```
//Discrete Time Markov Chain
dtmc

//variabile relativa al dado del giocatore 1
global dado1 : [0..6] init 0;

//variabile relativa al dado del giocatore 2
global dado2 : [0..6] init 0;

//variabile vincitore:    0 = non determinato;
//                  1 = vince giocatore 1;
//                  2 = vince giocatore 2;

global vincitore : [0..2] init 0;


//MODULO CHE GESTISCE LO SVOLGIMENTO DEL GIOCO
module Partita

//Lancio del primo dado (TRANSIZIONE)
[]     ( vincitore = 0 ) &
       ( dado1 = 0 ) &
       ( dado2 = 0 ) ->   1/6 : (dado1' = 1) +
                                1/6 : (dado1' = 2) +
                                1/6 : (dado1' = 3) +
                                1/6 : (dado1' = 4) +
                                1/6 : (dado1' = 5) +
                                1/6 : (dado1' = 6) ;

//Lancio del secondo dado (TRANSIZIONE)
[]     ( vincitore = 0 ) &
       ( dado1 != 0 ) &
       ( dado2 = 0 ) ->   1/6 : (dado2' = 1) +
                                1/6 : (dado2' = 2) +
                                1/6 : (dado2' = 3) +
                                1/6 : (dado2' = 4) +
                                1/6 : (dado2' = 5) +
                                1/6 : (dado2' = 6) ;

//PAREGGIO. SI GIOCA DI NUOVO (TRANSIZIONE)
[] ( vincitore = 0 ) &
   ( dado1 != 0 ) &
   ( dado2 != 0 ) &
   ( dado1 = dado2 ) ->  (vincitore' = 0) &
                               (dado1' = 0) &
                               (dado2' = 0);

//VINCE GIOCATORE 1 (TRANSIZIONE)
[] ( vincitore = 0 ) &
   ( dado1 != 0 ) &
   ( dado2 != 0 ) &
   ( dado1 > dado2 ) -> (vincitore' = 1);

//VINCE GIOCATORE 2 (TRANSIZIONE)
[] ( vincitore = 0 ) &
   ( dado1 != 0 ) &
   ( dado2 != 0 ) &
   ( dado1 < dado2 ) -> (vincitore' = 2);

endmodule
```

Figura 4.1: codice sorgente di un modello di gioco a titolo d'esempio

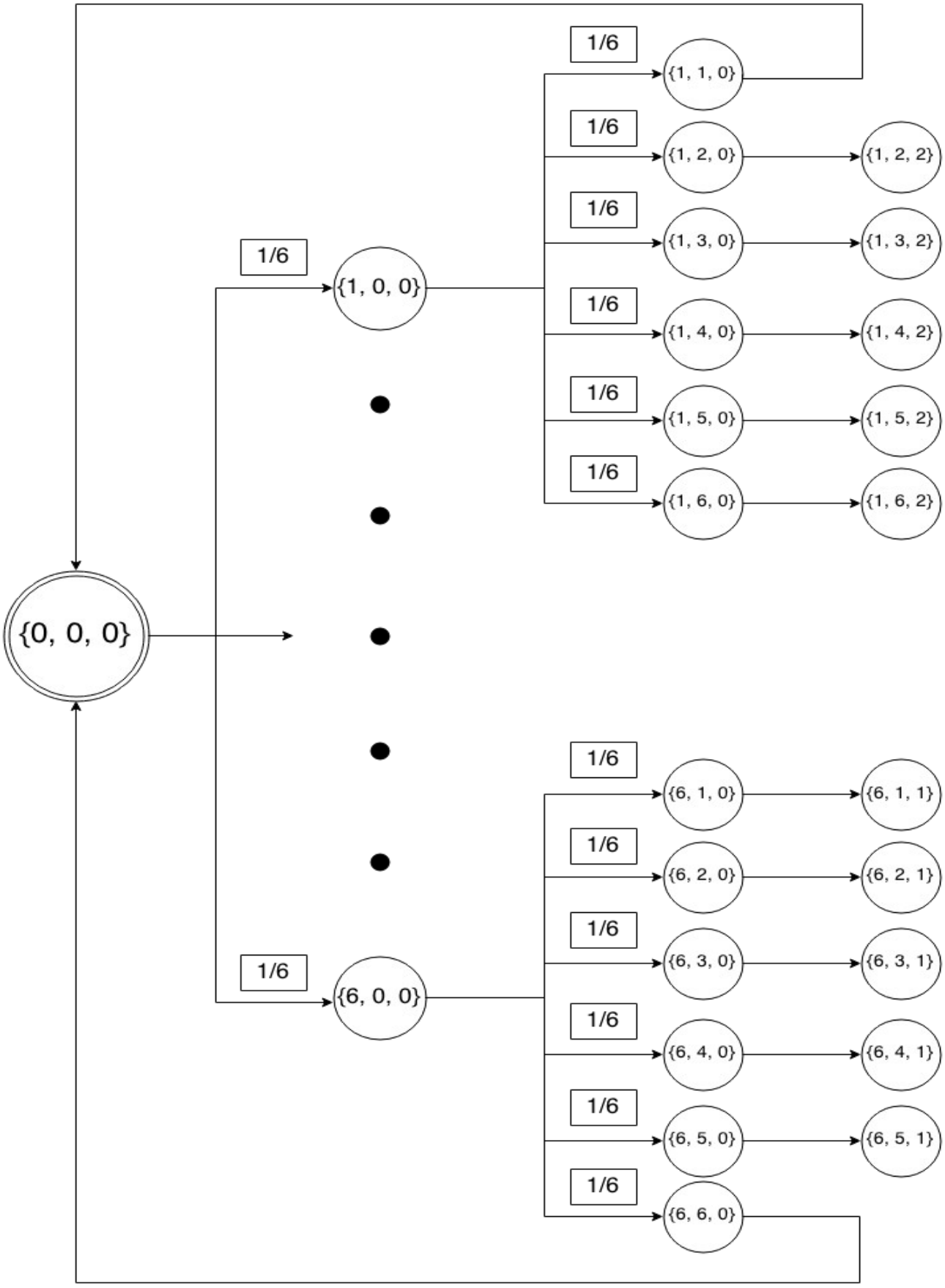


Figura 4.2: grafo con gli stati, le transizioni e le probabilità delle transizioni del modello di gioco

4.4 Le proprietà di un modello

Per verificare se un modello soddisfa o meno certe proprietà è possibile utilizzare strumenti di *model checking:* affinché il model checker interpreti e successivamente testi tali proprietà su un modello è necessario che queste siano espresse tramite una logica temporale adeguata al tipo di modello stesso. Per l'esempio che abbiamo mostrato e per i modelli implementati per i casi di studio che verranno presentati successivamente, la logica temporale adottata prende il nome di PCTL (Probabilistic Computation Tree Logic) [25] ed è l'estensione probabilistica della logica temporale CTL. Di seguito si propone una breve rassegna di alcuni degli operatori logici di maggiore interesse supportati dalla PCTL mostrandone l'applicazione in vari test.

A seconda del tipo di modello e soprattutto della funzione che questo è chiamato a simulare si renderà utile studiare una certa proprietà invece di un'altra.Tornando al modello di gioco presentato in questo capitolo può essere interessante riscontrare con quale probabilità la partita giunge a termine: utilizzando l'operatore logico **F**, anche chiamato "Future", si può chiedere al model checker di verificare con che probablità, partendo da un determinato stato (in questo caso si tratta di quello iniziale S0 = {0, 0, 0}), è possibile raggiungere uno degli stati in cui un giocatore ha ottenuto un punteggio più alto dell'avversario aggiudicandosi così la vittoria. In PRISM è possibile esprimere tale proprietà con la proposizione:

- P = ? [**F** (vincitore > 0)]

che significa letteralmente:

- "Con quale probabilità si raggiunge uno stato del modello nel quale la variabile vincitore assume un valore strettamente maggiore di zero?"

Il model checker, dopo aver costruito l'intera tabella composta da tutti i possibili stati e le possibili

transizioni corrispondente al modello, verificato la raggiungibilità degli stati e calcolato le probabilità legate alle varie transizioni, restituirà il risultato finale aprendo una finestra come quella che viene mostrata qui di seguito (nel caso in cui si utilizzi la GUI di PRISM):

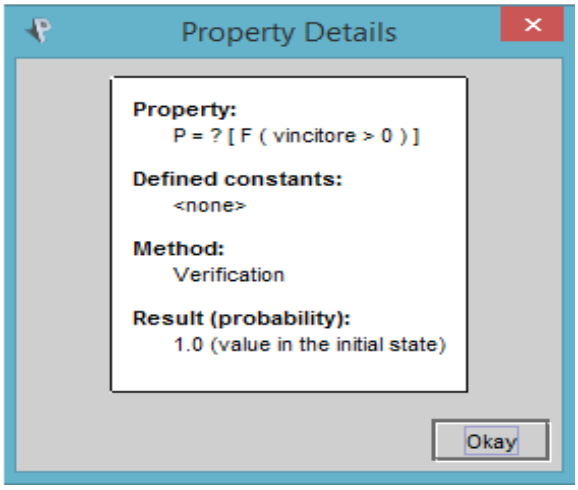


Figura 4.3: finestra che mostra il risultato di una verifica[11]

La finestra contiene la proprietà che si è deciso di verificare, l'elenco delle eventuali costanti definite (di queste si parlerà poco più avanti), il metodo formale utilizzato per la verifica e infine il risultato del test che per la proprietà sopra specificata è pari a uno.

L'operatore logico **G,** che sta per "Globally", a differenza di **F,** serve per esprimere l'eventualità che una certa proprietà sia valida in tutti gli stati del modello. Come riprova del test appena fatto, si può verificare allora quanto sia probabile che globalmente la variabile *vincitore* assuma un valore pari a zero, ovvero che non venga mai decretato un vincitore e che quindi il gioco non giunga a termine:

- P = ? [**G** (vincitore = 0)]

Se la verifica di questa proprietà desse un risultato maggiore di zero significherebbe che la struttura del modello contiene al suo interno un *loop* infinito, ovvero che in uno dei cammini possibili i giocatori lanciando il dado ottengono sempre risultato identico e sono quindi costretti a lanciare

11 Questa figura insieme alle altre relative alle proprietà espresse e verificate nell'esempio sono prese dalla GUI dell'ultima versione PRISM 4.2.beta1

nuovamente infinite volte.

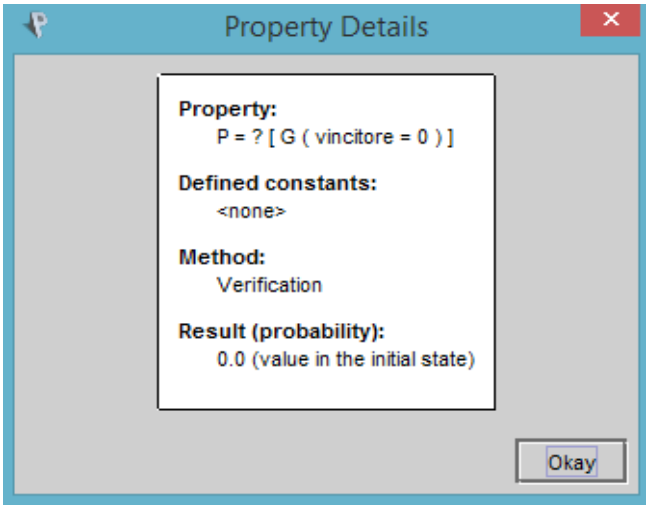


Figura 4.4: finestra che mostra il risultato di una verifica

Più precisamente: il modello che stiamo prendendo in considerazione contiene effettivamente tale *loop*, ma la probabilità che il sistema vi entri tende a zero, pertanto il risultato della verifica sarà appunto zero. Questo capita frequentemente all'interno dei modelli di giochi e pertanto è di estrema utilità assicurarsi che il sistema non possa mai finire in "circolo vizioso" del genere.

Nelle tabelle seguenti si mostrano altre proprietà che si è scelto di verificare e il risultato dei vari test:

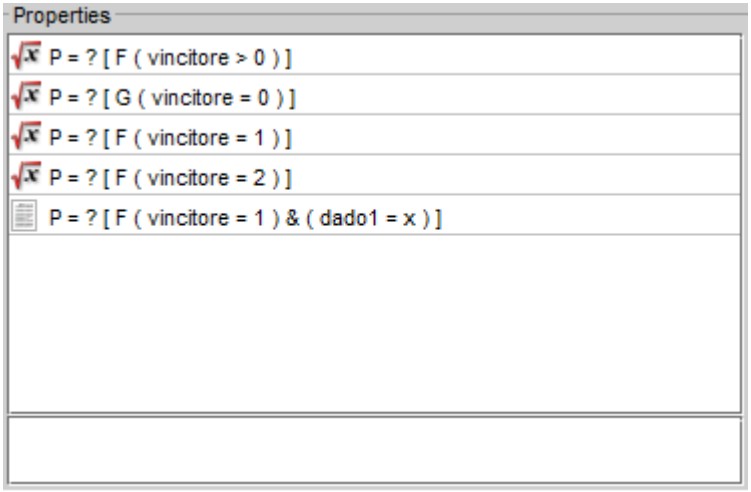


Figura 4.5: finestra che mostra la lista delle proprietà che si intende verificare

- Risultati

PROPRIETA'	RISULTATO
Vince il giocatore 1	0,499
Vince il giocatore 2	0,499

Tabella 4.1: risultati di due proprietà sopra indicate

La vittoria di uno dei giocatori risulta essere equiprobabile; i risultati non sono esattamente uguali a 0,5 come ci si aspetterebbe poiché durante il calcolo è possibile che per questioni di approssimazione si vadano "perdendo" delle cifre decimali.

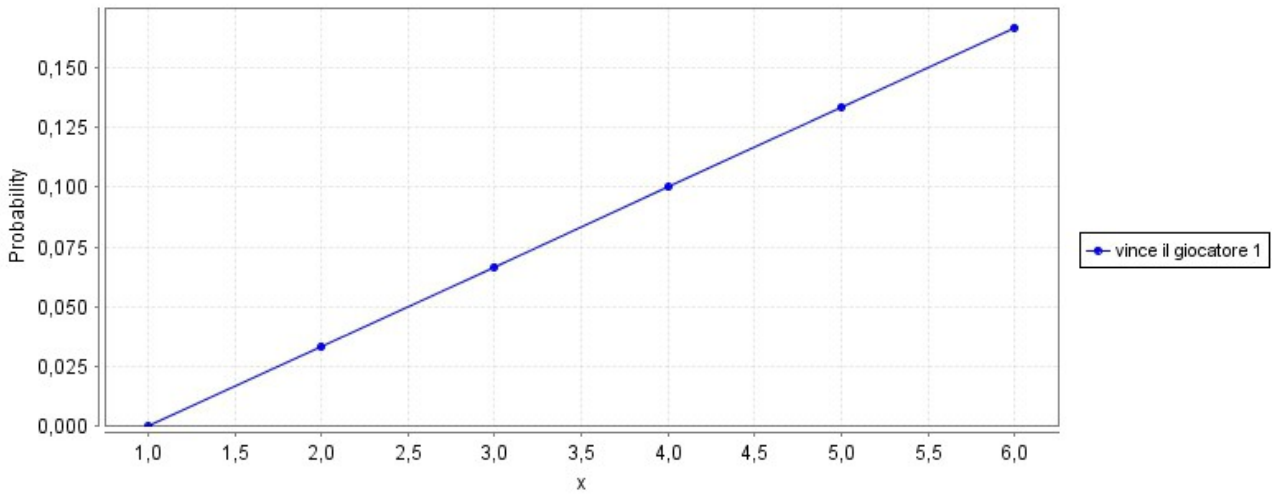
Per quanto riguarda l'ultima proprietà espressa nel riquadro mostrato in precedenza

- P = ? [(vincitore = 1) & (dado1 = x)]

questa contiene la costante *x* e sta a significare:

- "Quale è la probabilità che il giocatore Uno vinca ottenendo come risultato del lancio un valore pari a *x*?"

PRISM infatti permette di dichiarare delle costanti e, se necessario, di lasciarne indefinito il valore per poi impostarlo in fase di model checking. Questa possibilità si dimostra utile qualora per esempio si intendesse valutare il variare del risultato di una verifica al variare del valore di una data costante, sia essa coinvolta direttamente nel funzionamento del sistema, oppure utilizzata semplicemente come parametro di confronto. Dichiarando la costante *x* e lasciandone indefinito il valore, si può quindi verificare con quale probabilità il giocatore Uno si aggiudica la vittoria a seconda del risultato del suo lancio. Di seguito si mostra il grafico che rappresenta i risultati di tale test:



Figur 4.6: probabilità di vincere per il giocatore x, dato il risultato del lancio del dado

Sull'asse delle ascisse sono riportati i valori relativi alla costante *x*. Sull'asse delle ordinate si legge invece il risultato del relativo al test, ovvero la probabilità legata al risultato ottenuto con il lancio del dado dal giocatore uno. Si può notare che ottenendo uno la probabilità di vincere da parte del giocatore Uno sia pari a zero: infatti le uniche due eventualità possibili sono il pareggio, oppure la vittoria di Due che in tutti gli altri casi otterrebbe un punteggio più alto di Uno. La somma dei vari risultati riportati sul piano cartesiano è pari alla probabilità che globalmente possiede il giocatore Uno di vincere, ossia 0,5.

Un altro operatore logico che qui si intende mostrare è **U**, il quale sta per "Until". Grazie a questo operatore binario è possibile riscontrare se in un qualche stato di un possibile cammino una data proprietà sia vera, e in tutti gli stati precedenti sia vera un'altra proprietà. Un esempio può chiarire il concetto:

- P = ? [(*prop1*) U (*prop2*)]

significa:

- "Quale è la probabilità che *prop2* sia vera in un qualche stato di un cammino, e che *prop1* sia vera in tutti gli stati precedenti?"

Ci si aspetterebbe, vista la struttura del modello, che prima che venga sancito un vincitore, la variabile *vincitore* non prenda mai valori maggiori di zero. Questo significherebbe che ogni partita si svolge correttamente e che fintanto che uno dei due giocatori non supera l'altro con il suo punteggio si continuerà a lanciare i dadi. Allora la proposizione che esprime tale proprietà sarà:

- P = ? [(vincitore = 0) U (vincitore > 0)]

Il risultato di questa verifica (ovvia) è uno, ovvero il sistema funziona correttamente e le partite si svolgono senza intoppi.

Per una questione di semplicità ci limitiamo a citare soltanto questi operatori. Un elenco completo si trova nel manuale messo a disposizione dagli sviluppatori di PRISM [26].

5 CASI DI STUDIO

5.1 Inserimento di un *malus* in una dinamica completamente aleatoria

Figura 5.1.1: tabellone del Gioco dell'Oca illustrato

Il modello utilizzato in questo esperimento si ispira al celeberrimo Gioco dell'Oca. Di questo famoso gioco da tavola, nella versione moderna con il percorso che forma una spirale, si hanno notizie a partire dalla seconda metà del XVI secolo. Le regole del gioco impongono che a determinare il vincitore sia solo ed esclusivamente la sorte: siamo nell'ambito del *ludus*, e l'equilibrio insito nel gioco stesso è senza dubbio spostato sul polo dell'*alea*. La competizione tra i giocatori non riguarderà in nessun modo le loro capacità o il profitto che essi sono in grado di trarne, ma soltanto chi sarà fortunato al lancio del dado.

Nella sua versione originale il gioco si presenta con un tabellone composto da sessantatré caselle numerate progressivamente sulle quali le pedine, che rappresentano i giocatori, si muovono coprendo di volta in volta l'esatto numero di caselle indicato dal risultato del lancio del dado. Vi sono però alcune caselle per così dire speciali, sulle quali chi vi capita è costretto a eseguire alcune azioni particolari: disposte lungo il percorso esistono caselle di penalità che rimandano indietro il malcapitato giocatore di un certo numero di posizioni, caselle "oca" che lo proiettano in avanti e caselle che invece lo trattengono fermo per uno o più turni. Lo scopo del gioco è quello di arrivare per primi in fondo al tabellone, ma per farlo bisogna finire con un lancio esattamente sulla casella finale, poiché se la si supera questa ha l'effetto di far "rimbalzare" le pedine e farle andare a ritroso di un numero di posizioni pari alla differenza tra il risultato del lancio del dado e le caselle che dividono il giocatore dall'ultima (per esempio: se mi trovo sulla casella numero 60 e lanciando il dado totalizzo 5, "rimbalzerò" sulla casella finale ritornando alla casella numero 61).
La scelta che abbiamo fatto durante l'implementazione di questo modello ci ha portato però a semplificare ulteriormente il gioco e a mantenere soltanto alcune di queste caratteristiche.

5.1.1 *Struttura del modello*

All'interno del modello (DTMC) da noi costruito i giocatori, che sono denominati con le lettere *x* e *y,* si muovono dopo aver lanciato un dado a sei facce alternandosi nella turnazione. Lungo il tabellone questi incontrano soltanto un tipo di casella speciale, ossia la casella penalizzante che rimanda indietro di un numero predefinito di caselle. Rimane inalterata la funzione della casella finale. Sostanzialmente il modello mantiene inalterata la meccanica principale dell'originale (muoversi lungo il percorso), eliminando le caratteristiche che non erano strettamente necessarie ai fini dell'esperimento.

5.1.2 *Oggetto del caso di studio*

Come influisce sul bilanciamento del gioco il posizionamento di una casella di penalità? Le probabilità di vittoria dei giocatori vengono influenzate? Il distacco tra i giocatori subisce variazioni significative a seconda della posizione della casella? È vero che chi vince è finito mediamente meno volte sulla casella penalità?

Il lavoro di sperimentazione svolto tramite PRISM sul modello appena descritto del Gioco dell'Oca intende fornire un approccio metodologico che renda una scelta di design, come per esempio quella di inserire una casella speciale, il più efficace possibile. Questo caso di studio è infatti finalizzato all'individuazione di un numero ragionevole di caselle speciali di un certo tipo da inserire all'interno di un modello di gioco la cui meccanica principale è l'avanzamento delle pedine lungo un percorso. Con ragionevole si intende un numero di caselle che permetta di massimizzare gli effetti che si desidera che queste abbiano sul game-play.

5.1.3 *Esperimento*

Su un tabellone di sessantatré caselle, come nel gioco originale, si posiziona una casella penalità *p* che rimanda indietro il giocatore che ci finisce sopra di *n* caselle. La casella in questione presenta un'ulteriore caratteristica, ovvero non può rimandare indietro uno stesso giocatore più di *m* volte.
All'interno del codice che descrive il modello del gioco abbiamo inserito due variabili di tipo *integer* con funzione di contatore: *cpx* (**c**ontatore di **p**enalità di ***x***) e *cpy* (**c**ontatore di **p**enalità di ***y***) rispettivamente per il giocatore *x* e per il giocatore *y*. Queste variabili possono assumere valori [0, *m*] e sono inizializzate a *m*. Ogni volta che un giocatore finisce su *p* alla variabile contatore corrispondente viene sottratta un'unità. Una volta azzerato il contatore l'effetto che si otterrà sarà quello di *neutralizzare* l'effetto penalizzante di *p*, ovvero la disattivazione di *p* stessa. Sono necessarie due precisazioni riguardo alla scelta fatta di utilizzare un contatore in grado di contare solamente fino a *m* e di iniziare a contare proprio da *m*. Per quanto concerne la prima scelta, questa si è resa necessaria affinché si potesse evitare il caso in cui un giocatore, una volta finito su *p* continuasse a finirci all'infinito creando un numero di stati appunto infinito e impossibile da gestire. Riguardo alla seconda invece, è una scelta di stile: in questo modo infatti risulta più pratico modificare il valore di *m* dato che lo si deve fare una sola volta all'interno del codice (ovvero dove viene dichiarata e inizializzata la variabile contatore).

5.1.4 *Prima fase dell'esperimento: studio del "transito" su p*

Durante la fase iniziale si è deciso di osservare la probabilità dei giocatori *x* e *y* di finire su *p*. Per farlo abbiamo inizializzato

- $n = 0$

così da annullare l'effetto penalizzante di *p* stessa. In questo modo il giocatore che finisce su *p*, non

dovendo retrocedere, al turno successivo dovrà inevitabilmente avanzare escludendo quindi la possibilità di cadere nuovamente sulla casella penalizzante. Una casella con queste caratteristiche permette di vedere il problema come un problema binario: o il giocatore finisce una sola volta su *p*, o non ci finisce affatto. Ciò corrisponde, in altre parole ancora, a verificare la probabilità che i giocatori finiscano su una qualsiasi casella normale compresa nel tabellone di gioco. In base al transito dei giocatori su *p*, quindi dalla distribuzione di $P(p_x)$, si può pensare di decidere il posizionamento della casella penalizzante in base all'obiettivo che più ci interessa. Il primo tentativo è stato quello di verificare se il giocatore che vince è anche quello che mediamente finisce meno volte sulla casella penalizzante. Quindi avremo in totale 4 diversi casi:

- Entrambi i giocatori finiscono su *p* ($X \wedge Y$)
- Soltanto *y* finisce su *p* ($\neg X \wedge Y$)
- Soltanto *x* finisce su *p* ($X \wedge \neg Y$)
- Nessuno dei giocatori finisce su *p* ($\neg X \wedge \neg Y$)

Partendo dall'osservazione del giocatore *x* si può chiedere al model checker di restituirci la probabilità che *x* vinca, ovvero che completi il percorso (arrivando ad assumere un valore pari a 63) e che quindi passi inevitabilmente, posandovici sopra o schivandola, la casella *p:*

- $P(x = 63) \approx 0.53$[12]

Osservare la porzione dello spazio di probabilità corrispondente a tutte le partite vinte da *x* ci sarà utile per eseguire un confronto con i risultati dei test svolti più avanti. All'interno di questo spazio di possibilità è stato quindi monitorato il variare dei quattro casi sopra elencati al variare della casella sulla quale viene posizionata *p.* Di seguito si riporta una tabella con i risultati dei vari test:

12 La probabilità di vittoria del giocatore *x* è maggiore di 0,5 dal momento che questi muove per primo; questo svantaggio da parte di *y* si traduce quinid in una porzione di possibili cammini in cui a vincere è appunto *x*.

p	$X \wedge Y$	$\neg X \wedge Y$	$X \wedge \neg Y$	$\neg X \wedge \neg Y$
10	0.04	0.11	0.10	0.27
20	0.04	0.11	0.10	0.27
30	0.04	0.11	0.10	0.27
40	0.04	0.11	0.10	0.27
50	0.04	0.10	0.10	0.28
60	0.04	0.08	0.11	0.29

Tabella 5.1.1: risultati relativi al transito dei giocatori su p data la vittoria di x

Si nota che è molto più probabile che entrambi i giocatori non passino affatto su *p*, rispetto all'evenienza in cui entrambi i giocatori finiscano almeno una volta sulla casella, ma soprattutto che la probabilità che soltanto uno dei due ci finisca è in sostanza la stessa. Le uniche differenze si notano nell'ultima riga della tabella e sono dovute all'esistenza di partite in cui il giocatore *x* vince e il giocatore *y* non riesce neanche ad arrivare alla casella 60. Questo è sottolineato dai due record relativi al passaggio di un giocatore soltanto su *p* ($\neg X \wedge Y$ e $X \wedge \neg Y$)e dalla distribuzione di probabilità della casella di arrivo di *y* al termine della partita, data la vittoria di *x* (grafico qui di seguito):

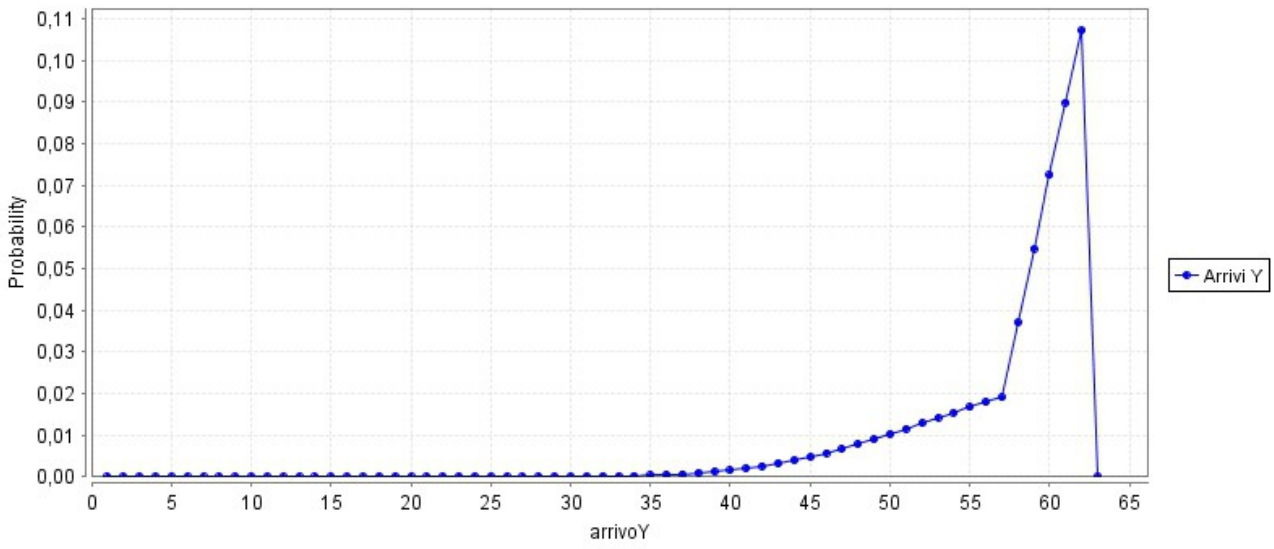


Figura 5.1.2: probabilità della casella di arrivo di y al termine della partita, data la vittoria di x

Il transito dei giocatori su *p* sembra rimanere stabile e bilanciato lungo tutto il percorso. Può essere ragionevole allora pensare di ripetere le verifiche dando un peso a *p*, quindi aumentando *n*, per capire se l'effetto di rimandare indietro i giocatori interferisce sul bilanciamento generale (probabilità di vincere), sul transito dei giocatori su *p* e sulla combinazione delle due.

Si è optato per infliggere una penalità di 6 caselle, utilizzando le stesse posizioni per *p* adottate nel test precedente (prima colonna di sinistra). Di seguito i risultati delle tre diverse serie di dati: in blu i risultati relativi alle partite in cui i due giocatori finiscono lo stesso numero di volte su *p,* in verde quelli relativi alle partite in cui è il giocatore *y* a finirci più volte e in rossa l'eventualità opposta (leggendo le proprietà riportate nelle intestazioni delle colonne si deve tener presente che se il contatore di un giocatore ha un valore maggiore di quello dell'avversario significa che questi è capitato meno volte su *p*):

p	P [F (x = 63) & (cpx = cpy)]	P [F (x = 63) & (cpx > cpy)]	P [F (x = 63) & (cpx < cpy)]
10	0,28	0,15	0,08
20	0,29	0,15	0,08
30	0,29	0,15	0,08
40	0,29	0,15	0,08
50	0,3	0,14	0,08
60	0,31	0,12	0,09

Tabella 5.1.2: risultati relativi alla probabilità di vittoria di x e all'influenza della casella penalizzante (n = 6)

Grafico dell'esperimento relativo ai risultati aggregati:

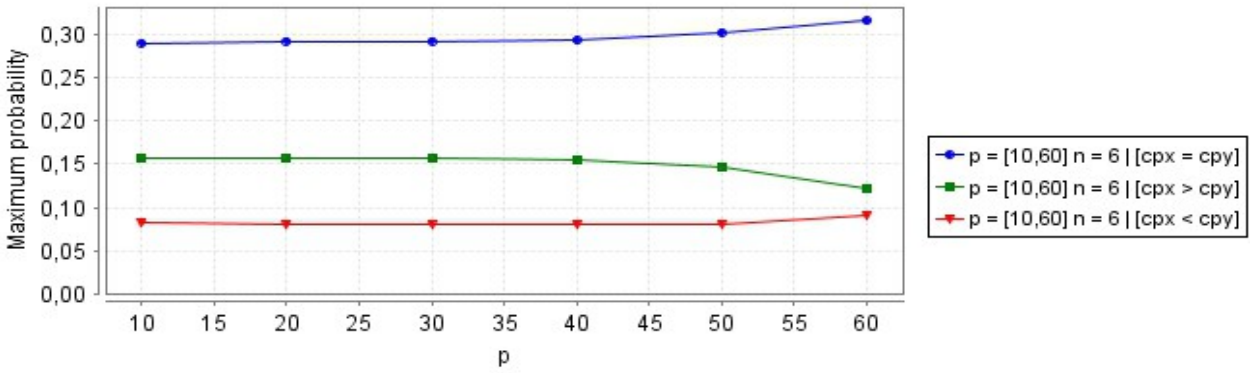


Figura 5.1.3: grafico che riporta i risultati mostrati nella tabella precedente

Come ultimo test si è osservata la probabilità che *x* vinca al variare del posizionamento di *p*:

p	P[F (x = 63)]
10	0,529
20	0,529
30	0,529
40	0,529
50	0,529
60	0,528

Tabella 5.1.3: probabilità di vittoria di x relativamente alla posizione di p

Come si può notare dalla tabella sopra riportata la probabilità che il giocatore *x* vinca rimane invariata indipendentemente dal posizionamento *p*. L'unica differenza si nota nell'ultima riga ed è legata al fatto che, essendo *x* il giocatore che muove per primo, sarà lui stesso ad essere leggermente più penalizzato da *p* se posizionata sulla casella 60, dato che più facilmente sarà lui a finirci per primo permettendo in questo modo a *y* di recuperare lo svantaggio e vincere. Il numero di partite possibili in cui ciò accade corrisponde quindi a quel sensibile abbassamento riscontrabile anche negli altri esperimenti in corrispondenza appunto del posizionamento di *p* sulle caselle intorno alla numero 60. Si può vedere anche come la probabilità che chi vince (in questo caso *x*) passi meno

volte su *p* è generalmente più alta. Questo si riscontra anche se si raggruppano le osservazioni aggregando i risultati e calcolandone una media:

	Media dei risultati
P[F (x = 63) & (cpx = cpy)]	0,29
P[F (x = 63) & (cpx > cpy)]	0,14
P[F (x = 63) & (cpx < cpy)]	0,08

Tabella 5.1.4: media dei risultati di cui alla tabella 5.1.2

L'esperimento è ripetibile con qualsiasi valore di *p* e dell'ammontare della penalità *n*.

5.1.5 *Posizionamento di più caselle penalizzanti*

Come per il test svolto in precedenza, anche per questi esperimenti si è andati a osservare la frazione di cammini corrispondenti alle partite vinte da *x*. Per ora si è visto che mediamente, chi finisce meno volte sopra una casella penalizzante che abbia le caratteristiche di *p* sarà portato a vincere più spesso, anche se di poco. Nei test seguenti andremo a osservare se ciò rimane vero anche quando le caselle penalizzanti sono più di una e disseminate lungo tutto il tabellone di gioco. Andremo a osservare ancora una volta lo spazio di possibilità legato alla vittoria del giocatore *x*.

L'esperimento inizialmente consiste nell'inserimento di quattro caselle penalizzanti; successivamente le caselle diventeranno cinque e infine sei. Le caselle sono state posizionate a intervalli regolari e l'ammontare della penalità fissato a sei caselle, come in precedenza. I contatori *cpx* e *cpy* per questi esperimenti sono stati invece inizializzati a 10, differentemente da come era stato fatto nei test precedenti.

- 4 caselle penalità

$P = \{15, 30, 45, 60\}$	Risultato
CPX = CPY	0,12
CPX > CPY	0,27
CPX < CPY	0,13

- 5 caselle penalità

$P = \{15, 25, 35, 45, 55\}$	Risultato
CPX = CPY	0,1
CPX > CPY	0,3
CPX < CPY	0,125

- 6 caselle penalità

P = {10, 20, 30,40, 50, 60}	Risultato
CPX = CPY	0,092
CPX > CPY	0,303
CPX < CPY	0,125

Tabella 5.1.5: risultati relativi alla probabilità di vittoria di x al variare del numero di caselle penalizzanti

La probabilità che *x* vinca rimane invariata nei tre casi e pari a

- P = 0,52[13]

Infine potrebbe essere utile inserire una quantità nettamente superiore di caselle penalizzanti per vedere se i risultati sono differenti e di quanto. Quindi andiamo a posizionare dieci caselle disseminandole lungo tutto il tabellone.

- 10 caselle penalità:

P = {7, 14, 21 ,28 , 35, 42, 49, 56, 60, 62}	Risultato
CPX = CPY	0,09
CPX > CPY	0,302
CPX < CPY	0,124

Tabella 5.1.6: risultati relativi alla probabilità di vittoria di x con 10 caselle penalizzate sparse lungo il tabellone

È da notare come la probabilità che *x* vinca sia ulteriormente diminuita in questo ultimo esperimento, scendendo leggermente a

- P = 0,517

e questo è dovuto al motivo di cui sopra, ovvero alla presenza di caselle che penalizzano più *x* di *y*.

13 Relativamente al secondo caso, in cui sono state posizionate 5 caselle penalizzanti, la probabilità di vittoria di *x* somma a 0.525: un piccolo innalzamento dovuto alla “lontananza” dell'ultima casella speciale dal traguardo.

Indipendentemente da dove sono posizionate, all'aumentare delle caselle penalizzanti aumenta anche la probabilità che chi vince sia anche il giocatore che ci è capitato di meno. Da uno 0,15 iniziale, ricavato testando un tabellone con una sola casella penalizzante posta su sei diverse posizioni (questo valore è sempre da intendere in riferimento allo spazio di probabilità definito dai soli cammini che terminano con *x* vincitore), si passa a valori che si attestano attorno al doppio esatto (nelle tabelle sono evidenziati di rosso). Date le caratteristiche del tabellone di gioco (63 caselle) e di *p,* relativamente alle partite in cui *y* finisce più volte sopra le caselle speciali, si nota che la probabilità di vittoria (in questo caso di *x*) aumenta all'aumentare del numero di caselle penalizzanti, e si distribuisce in modo stabile con l'inserimento di 5 o più caselle. Da aggiungere inoltre l'oscillazione osservata in seguito al posizionamento di *p* su caselle vicine al traguardo.

Andamento di P [(*x*=63) & (cp*x*>cp*y*)] all'aumentare delle caselle di penalità:

(X vince) & (CPX > CPY)	Risultato
1 casella penalità	0,14
4 caselle penalità	0,275
5 caselle penalità	0,301
6 caselle penalità	0,303

Tabella 5.1.7: riassunto relativo ai soli casi in cui x è finito meno volte sulle varie caselle penalizzanti

5.1.6 *Conclusioni*

Compiere delle scelte implementative spesso richiede un certo numero di verifiche volte a saggiare gli effetti che queste scelte possono avere sul gioco che stiamo progettando. I test svolti hanno principalmente lo scopo di sondare un possibile approccio al design di un gioco la cui meccanica principale è l'avanzamento sulla plancia di gioco delle pedine determinato dal lancio di un dado a sei facce.

Nella prima parte di questo caso di studio si è cercato di affrontare il posizionamento di una casella penalizzante all'interno di un modello del *Gioco dell'Oca*, andando a osservare gli effetti che questa può avere su alcuni aspetti del gioco stesso. Per prima cosa ci si è chiesti dove è opportuno che una casella con le stesse caratteristiche di quella utilizzata nei test possa essere posizionata all'interno del tabellone di gioco. In particolare si è visto che, testando posizioni diverse, i risultati non subiscono differenze sostanziali: la probabilità che i giocatori vi finiscano sopra si mantiene costante indipendentemente dalla casella sulla quale si posiziona la penalità. Quindi si è andati a osservare quanto e come incide effettivamente una penalità di un certo tipo sulla determinazione del vincitore e sul transito su di essa. Iniziando questo esperimento con una sola casella penalizzante si è notato che, come era legittimo aspettarsi, finendoci un numero minore di volte un giocatore ha una maggiore, seppur di poco, probabilità di vincere. Una volta verificata questa dinamica ci si è chiesti se questa potesse ripetersi anche qualora le caselle penalizzanti fossero state in numero maggiore: in particolare nei test successivi si è andati a vedere se esiste un numero ottimale di caselle con queste caratteristiche che, se posizionate lungo il tabellone, possa avere un qualche effetto rilevante sullo svolgimento del gioco. I risultati che si è scelto di mostrare sono soltanto quelli relativi ai test svolti con un numero di caselle compreso tra quattro e sei, più il test fatto con dieci caselle penalizzanti. Il motivo principale di questa scelta risiede nel fatto che le prime differenze rilevanti nei risultati si

iniziano a riscontrare soltanto una volta inserita anche la quarta casella penalizzante. L'effetto principale che un maggior numero di caselle penalità ha sulle dinamiche del gioco è quello di acuire il vantaggio che si acquisisce nello schivarle: la probabilità che chi si aggiudica la vittoria sia il giocatore che è riuscito a finire meno volte sulle caselle penalità[14] raddoppia rispetto a quella riscontrata nelle partite giocate su un tabellone con una sola di queste caselle. Dati i risultati dei quattro casi presi in considerazione, ovvero delle partite giocate con un tabellone che contiene una, poi quattro, cinque, sei e infine dieci caselle penalità, si può pensare che un numero ottimale di caselle di questo tipo può essere 5, ovvero il numero minimo che ne massimizza l'effetto.

14 In modo chiaramente fortuito dal momento che, come si è detto in apertura di questo caso di studio, il Gioco dell'Oca è unicamente basato sulla sua componente di alea

5.2 Studio di possibili strategie dominanti in un modello di gioco di strategia

Figura 5.2.1: plancia di gioco, armate e carte del gioco Risiko[15]

Risiko [4] è un gioco da tavolo pensato e realizzato dal regista francese Albert Lamorisse e che viene pubblicato per la prima volta nei primi anni sessanta. Originariamente il gioco si chiamava "La conquete du monde" ("La conquista del mondo") e prevedeva un solo obiettivo, ossia la conquista di tutti i territori rappresentati sul tabellone del gioco. Al regolamento originale nel corso degli anni sono state apportate delle modifiche che hanno reso Risiko uno dei giochi di simulazione di guerra tra i più giocati al mondo.

15 L'immagine è presa dal sito ufficiale di Risiko disponibile da: http://www.risiko.it/index.php

5.2.1 *Caratteristiche del gioco*

Risiko è un gioco di strategia nel quale due o più avversari cercano di raggiungere l'obiettivo che gli è stato assegnato casualmente all'inizio della partita. Solitamente gli obiettivi consistono nel conquistare un certo numero di territori, distruggere le armate di un avversario,conquistare determinati continenti e così via. La plancia sulla quale si gioca si presenta come una mappa geopolitica della Terra nella quale sono considerati adiacenti sia gli stati che confinano naturalmente sia quelli che, anche se divisi per esempio da tratti di mare, sono comunque collegati da linee tratteggiate. Ogni giocatore, oltre a ricevere un obiettivo personale da perseguire, viene anche dotato di un certo numero di armate con le quali occupa i territori a lui assegnati in partenza e quelli che successivamente conquisterà giocando. Chi perde tutte le proprie armate viene eliminato dal gioco. Il gioco finisce quando uno dei giocatori raggiunge il suo obiettivo personale.

5.2.2 *Modalità degli scontri*

Se un giocatore intende conquistare un territorio che è già occupato da un avversario deve prima eliminare tutte le armate che vi si trovano sopra: per farlo dovrà dichiarare quale è il territorio che intende attaccare (che deve essere necessariamente adiacente a uno dei territorio da lui occupati) e con quante armate intende farlo. Il giocatore che viene attaccato dovrà da parte sua dichiarare con quante armate vuole difendersi. Il numero di armate utilizzabili negli scontri va da un minimo di uno a un massimo di tre. Uno scontro si svolge interamente tramite il lancio di tanti dadi a sei facce quante sono le armate che si è deciso di impiegare nella battaglia. Una volta che chi attacca ha lanciato i suoi dadi, si lanciano i dadi della difesa: per determinare il vincitore si confronta il dado dell'attaccante che ha ottenuto il punteggio più alto con quello con il punteggio più alto del difensore. Nel caso i dadi lanciati siano più di uno per giocatore, si procede con il confronto

seguendo un ordine progressivo. Un esempio aiuterà a comprendere meglio:

- Giocatore Uno → Attacca con 2 armate;
- Giocatore Due → Difende con 2 armate;

Si procede al lancio dei dadi:

	DADO 1	DADO 2
ATTACCO	4	6
DIFESA	5	3

Il confronto sarà fatto in questo modo, mettendo i dadi di ogni giocatore in ordine decrescente di punteggio:

	DADO 2	DADO 1
ATTACCO	6	4
	↓	↓
	DADO 1	DADO 2
DIFESA	5	3

Tabella 5.2.1: esempio di scontro tra due giocatori che impiegano due armate ciascuno

Dal momento che entrambi i punteggi sono a favore dell'attacco il giocatore che si stava difendendo perderà due armate, le quali dovranno essere tolte dal territorio su cui si trovavano. In caso di pareggio invece la vittoria va alla difesa. Se in una battaglia il giocatore in difesa esaurisce le armate in suo possesso sul territorio coinvolto, questo viene occupato dall'altro che è chiamato a posizionarvi almeno una delle sue armate. Chi decide di attaccare un altro territorio può quindi farlo solo dopo essersi assicurato che sul territorio dal quale intende sferrare l'attacco rimanga presente almeno un'armata: questo per garantire che nel caso in cui l'attacco perda tutte le armate, almeno una rimarrà a presidiare il territorio. All'inizio di ogni turno ciascun giocatore riceve tante armate

quanti sono i territori da lui occupati diviso per tre (e arrotondato per difetto).

5.2.3 *Caratteristiche del modello*

Nel modello da noi implementato non tutte le caratteristiche del gioco originale sono state mantenute. Concentrandosi sulla meccanica principale (i combattimenti tra armate avversarie) si è scelto di rappresentare la plancia di gioco con un numero inferiore di territori: il modello creato ha un totale di 6 territori (a fronte dei 42 del gioco originale), ognuno corrispondente a una delle sfere sotto rappresentate, collegati da 8 *link* in modo da rendere possibile le interazioni tra alcuni di questi territori e da escluderne altre. Più precisamente i collegamenti servono a indicare quali territori possono essere coinvolti in una battaglia: se due territori sono adiacenti possono "duellare", altrimenti non possono. Lo schema qui mostrato ha lo scopo di dare una possibile disposizione topologica della plancia di gioco:

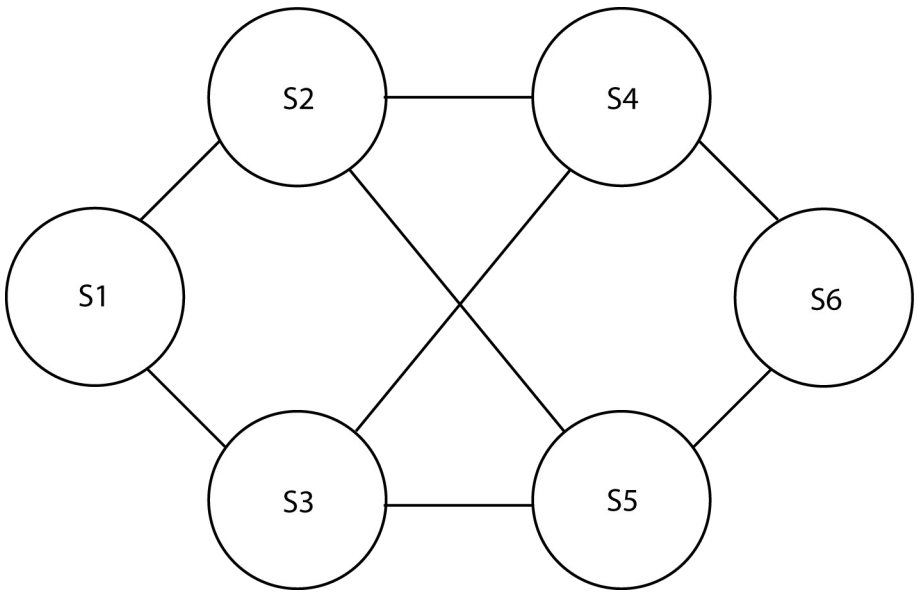


Figura 5.2.2: possibile disposizione topologica della plancia di gioco del modello

Riguardo alle armate da spostare in caso di conquista di un territorio si è deciso di fissarlo a uno. Invece per gli scontri tra le armate avversarie si è mantenuto il set di regole previsto per la versione

originale del gioco ma assumendo che a ogni attacco vengano schierate una sola armata per l'attacco e una per la difesa. Simulando quindi il lancio di un dado per giocatore, le probabilità di vincere da parte dell'attaccante e del difensore sommano a:

- P (vince l'attacco) = 0,417;
- P (vince la difesa) = 0,583;

Come si vedrà più avanti nel corso degli esperimenti queste due probabilità saranno "pesate", ovvero saranno moltiplicate per una costante in modo da simulare uno scontro in cui sono coinvolte più armate (il quale darebbe effettivamente più chance di vincere a chi ne impiega di più).
Nel modello sono previsti soltanto due giocatori che saranno rappresentati dalle variabili *giocatoreZero* e *giocatoreUno*.

5.2.4 *Analisi del modello*

È bene specificare prima che in questo caso di studio tutte le proprietà sono simulate e non verificate poiché la grandezza del modello e i limiti di memoria non permettono di svolgere verifiche in tempi ragionevoli. I risultati riportati sono quindi il frutto di un model checking statistico effettuato su un *sample* costituito da 10000 cammini scelti in maniera casuale.

Il numero delle armate iniziali relativamente a ogni singolo territorio è stato affidato in fase di implementazione a una costante per la quale non è stato specificato il valore: in questo modo è stato possibile determinarlo prima di iniziare ogni singola simulazione. Ad ogni modo, escluso per l'esperimento incentrato sulla distribuzione ponderata delle armate iniziali, per tutti gli altri che vedremo in questo caso di studio le armate iniziali disposte su ogni territorio saranno pari a 6.

Infine i territori saranno assegnati equamente tra i due giocatori che ne riceveranno quindi tre a testa: i tre territori di sinistra (S1, S2 e S3) al giocatore Zero e i tre di destra (S4, S5 e S6) al giocatore Uno.

5.2.5 *Studio delle proprietà del modello: la durata della partita e delle sue fasi*

L'esperimento condotto sulla durata della partita cerca di mostrare come varia la probabilità che una partita possa finire entro un certo numero di turni in base al variare dell'atteggiamento tenuto dai giocatori. La tipizzazione degli atteggiamenti da noi proposta è riassunta nella tabella seguente:

	PASSIVO	NEUTRO	AGGRESSIVO
SCELTA	Prob	Prob	Prob
ATTACCO	0,1	0,5	0,9
PASSO IL TURNO	0,9	0,5	0,1

Tabella 5.2.2: tipizzazione delle tattiche

Le probabilità associate all'attacco o al passare il turno all'avversario intendono simulare un atteggiamento:

- **Passivo**, che raramente si cimenta in uno scontro diretto con l'avversario
- **Neutro**, che in modo equiprobabile passa il turno o attacca un territorio dell'avversario
- **Aggressivo**, decisamente propenso ad attaccare direttamente l'avversario

Per praticità useremo queste tre definizioni ogni volta che ci riferiremo a una delle tattiche adottate dai giocatori nel modello. Assumendo come ragionevole un numero massimo di turni pari a 350, i dati di seguito riportati fanno riferimento a partite che si concludono entro questo limite.
Nel grafico che segue si mostrano i risultati relativi a partite nelle quali i due giocatori adottano la stessa tattica indicata nella legenda:

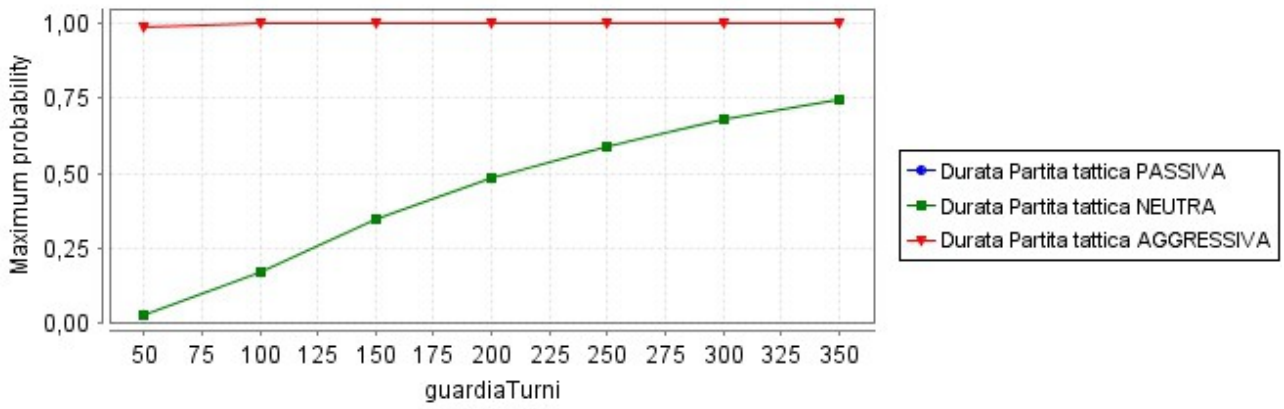


Figura 5.2.3: probabilità che la partita finisca entro un certo numero di turni

Si nota l'assenza della traccia relativa alle partite giocate con tattica Passiva. Questa è dovuta allo sbilanciamento della tattica stessa verso il passaggio del turno all'avversario, il che rende di fatto le partite infinite, facendo accumulare armate ai giocatori e non permettendo di farle rientrare entro il limite assunto di 350 turni.

Il grafico e la tabella che seguono intendono mostrare il monitoraggio della probabilità dei due giocatori di vincere quando adottano la stessa tattica: questo test serve per escludere che il giocatore Zero sia avvantaggiato a prescindere per il solo motivo che parte sempre per primo. La tabella mostra i risultati aggregati per partite che durano da un minimo di 0 a un massimo di 350 turni. I risultati sono calcolati su un totale di 10000 partite:

Tattica adottata	Prob ZERO vince	Prob UNO vince
Passiva	0	0
Neutra	0,377	0,373
Aggressiva	0,493	0,522

Tabella 5.2.3: probabilità di vittoria dei giocatori

Anche in questo grafico sono assenti i dati relativi alla tattica Passiva per il motivo già citato poco sopra. Il valore della variabile *p1* corrisponde alla probabilità con la quale il giocatore a ogni turno decide di attaccare l'avversario. Quando *p1* viene inizializzata a 0,5 la probabilità con la quale i giocatori passano il turno è identica. Quando vale 0,9 la probabilità che passino il turno è invece P = (1 – *p1*), ovvero 0,1.

5.2.6 *Differenziazione delle tattiche*

Abbiamo quindi differenziato le tattiche dei giocatori facendo in modo che sia il solo giocatore Zero ad adottare una tattica aggressiva e che il giocatore Uno invece ne adotti una neutra. Monitorando l'andamento delle probabilità di vittoria dei due giocatori si avrà che:

	AGGRESSIVA	NEUTRA
numero Turni	Prob ZERO vince	Prob UNO vince
50	0.4872	0.2254
100	0.6567	0.2675
150	0.7024	0.2921
200	0.7138	0.2957
250	0.7155	0.2901
300	0.7079	0.2896
350	0.7148	0.2837

Tabella 5.2.4: probabilità di vittoria dei giocatori relativamente al numero di turni giocati

Nel grafico si possono osservare i risultati della tabella appena mostrata:

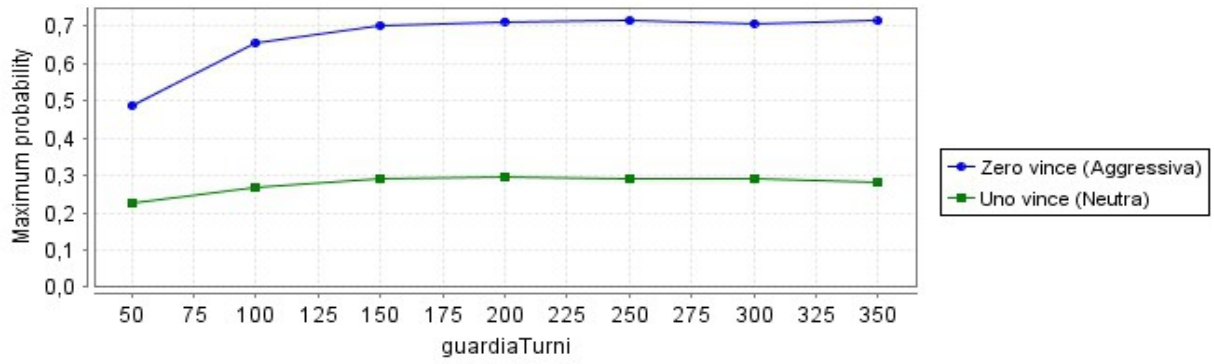


Figura 5.2.4: grafico che riporta i valori mostrati nella tabella precedente

Ripetiamo l'esperimento invertendo le tattiche adottate dai giocatori, per tentare di escludere il fattore di vantaggio che potrebbe avere chi fa la prima mossa. Le serie di dati riportati qui sotto in tabella sono relative alle probabilità di vincere dei due giocatori, tenendo conto che il giocatore Zero questa volta adotterà una tattica neutra , e il giocatore Uno invece adotterà la tattica aggressiva.

I colori utilizzati sono gli stessi di sopra:

	AGGRESSIVA	NEUTRA
numero Turni	Prob ZERO vince	Prob UNO vince
50	0.2495	0.4636
100	0.2934	0.6364
150	0.3041	0.6919
200	0.3042	0.6952
250	0.3021	0.7022
300	0.2941	0.6984

Tabella 5.2.5: probabilità di vittoria dei giocatori relativamente al numero di turni giocati

Quindi mostriamo il grafico:

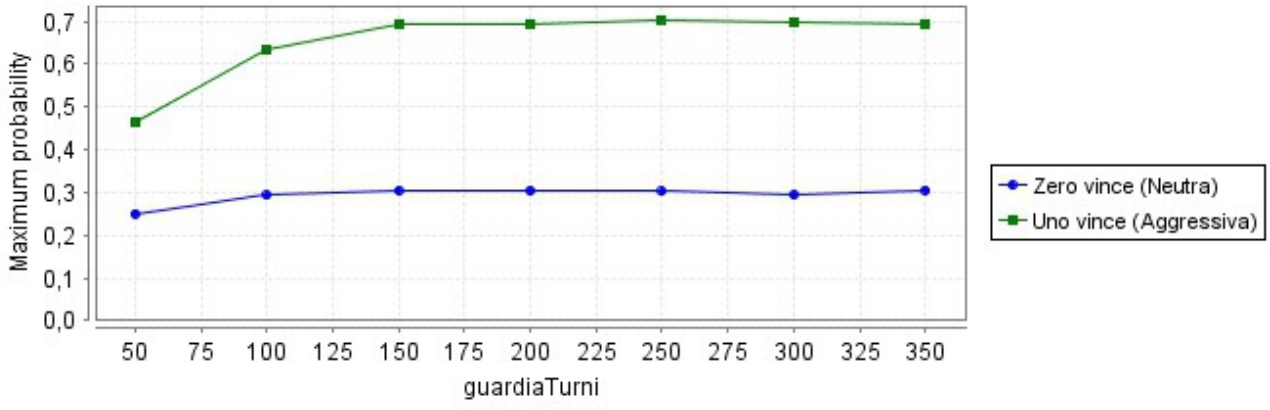


Figura 5.2.5: grafico che riporta i valori mostrati nella tabella precedente

Tralasciando per un istante l'aspetto della durata del gioco, si mostrano di seguito le probabilità di vittoria da parte dei giocatori relativamente a partite in cui Zero gioca in modo aggressivo mentre Uno adotta ogni volta una delle tre tattiche viste fin qui. Nella seconda tabella si mostrano i risultati ottenuti invertendo nuovamente gli approcci alla partita:

	AGGRESSIVA	
	Prob ZERO vince	Prob UNO vince
Passiva	0.970	0.030
Neutra	0.717	0.282
Aggressiva	0,488	0.519

		AGGRESSIVA
	Prob ZERO vince	Prob UNO vince
Passiva	0.032	0.969
Neutra	0,313	0.701
Aggressiva	0.485	0.517

Tabella 5.2.6: tabelle che riassumono le probabilità di vittoria dei singoli giocatori

Queste ultime due tabelle saranno utili nel confronto con le verifiche che verranno fatte più avanti. Riassumendo, se si assume che entrambi i giocatori adottino una tattica neutra durante tutta la partita occorreranno circa 200 turni affinché il 50% delle partite giocate si concluda con un vincitore. Le partite si accorciano drasticamente se entrambi i giocatori propendono decisamente verso un atteggiamento aggressivo, infatti dal grafico è evidente come sopra i 50 turni la probabilità che la partita termini si mantenga praticamente costante e pari a 1 (vedi figura 5.2.3). Se invece a giocare aggressivamente è soltanto uno mentre l'altro giocatore gioca senza nessuna tattica predominante (neutro), il primo vedrà le sue probabilità di vittoria alzarsi più rapidamente e mantenersi in modo costante decisamente più alte.

5.2.7 *Un ulteriore elemento strategico: una disposizione delle armate ponderata*

In fase di setup del gioco tutti i giocatori ricevono un numero predefinito di armate che sono chiamati a posizionare sui territori che vengono loro assegnati. Per questo esperimento le armate sono state predisposte nel seguente modo: il giocatore Zero avrà 6 armate per ogni territorio in suo possesso (totale 3 territori e 18 armate), mentre il giocatore Uno tenterà una disposizione sbilanciata con 8 armate sui territori più esposti e 2 soltanto sul territorio più riparato. L'immagine ha lo scopo di dare una possibile raffigurazione topologica della plancia dopo il setup appena descritto:

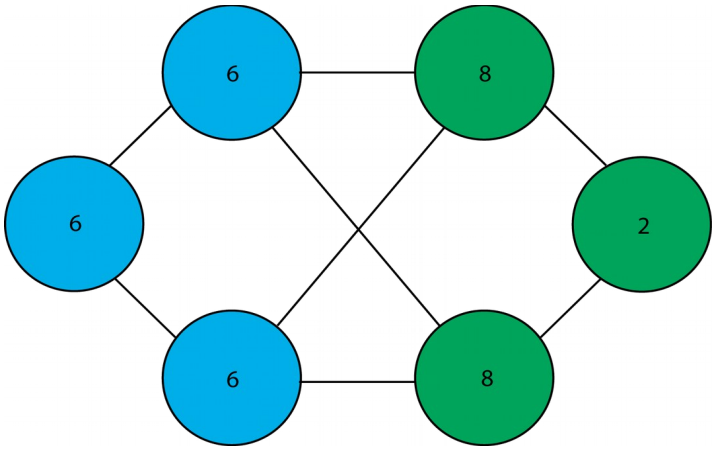


Figura 5.2.6: disposizione sbilanciata delle armate (barricamento)

I cerchi blu corrispondono ai territori in controllo del giocatore Zero e quelli verdi ai territori del giocatore Uno prima di iniziare a giocare. Il numero che contengono si riferisce al numero delle armate disposte su ogni territorio. Una tale disposizione delle armate è da considerarsi non solo come possibile disposizione iniziale dal momento che, durante le varie fasi del gioco, i giocatori continueranno a ricevere armate bonus che dovranno distribuire sui propri territori come meglio credono: una tattica che potremmo definire di **barricamento** è da considerarsi quindi praticabile in qualsiasi momento della partita.

Per prima cosa possiamo andare a vedere se e come questa disposizione, se adottata in partenza, vada a incidere sulla durata della partita. Assumendo che i due giocatori utilizzino la medesima tattica e scegliendo di utilizzare le stesse tattiche utilizzate negli esperimenti precedenti (ovvero una neutra, una aggressiva e una passiva), la durata della partita non subisce nessuna sostanziale modificazione. Di seguito si riporta solo il grafico:

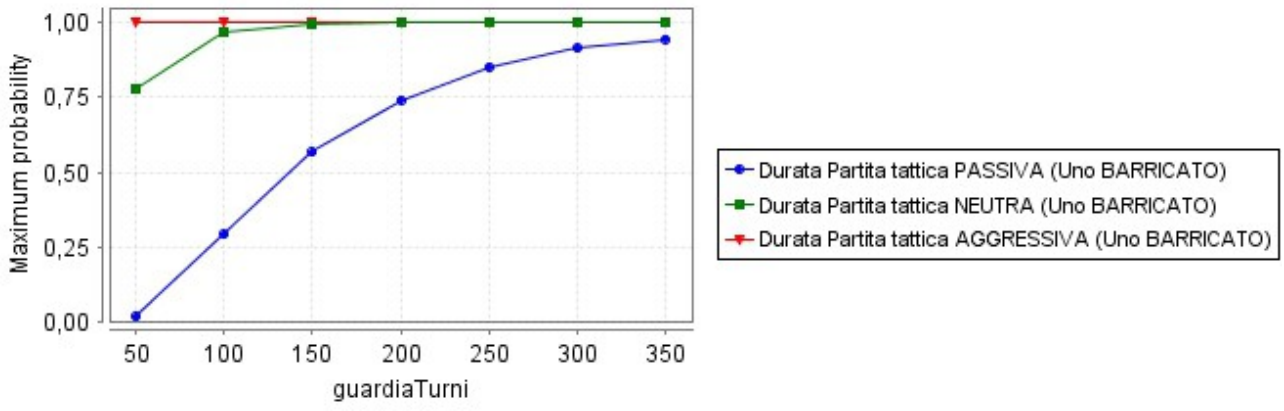


Figura 5.2.7: grafico che riporta i valori relativi alla probabilità che la partita termini entro un certo numero di turni, data la disposizione delle armate di Uno e facendo adottare a entrambi di volta in volta la stessa tattica

Ripetendo l'esperimento nel tentativo di monitorare l'andamento della probabilità di vittoria dei due giocatori si ottiene:

Tattica adottata	Prob ZERO vince	Prob UNO vince
passiva	0	0
neutra	0.375	0.636
aggressiva	0.424	0.576

Tabella 5.2.7: tabella che raccoglie i risultati contenuti nel grafico precedente

A parità di tattica adottata barricarsi disponendo più armate nei territori direttamente confinanti con quelli dell'avversario (che quindi possono essere attaccati e dai quali si può attaccare) sembra essere

più remunerativo, infatti la probabilità di vincere per chi sceglie piazzamento delle armate del genere è in ogni caso un po' più alta. La differenza è in parte dovuta al fatto che durante i primi scontri in cui è possibile coinvolgere soltanto i territori centrali (S2, S3, S4 e S5) si decide anche lo sviluppo successivo della partita: chi sarà in grado di sostenere più perdite inizialmente potrebbe ritrovarsi nella condizione favorevole di avere armate sufficienti per la conquista dei territori dell'avversario una volta che questi risulti più indebolito.

5.2.8 *Uno scontro più bilanciato: modulazione della probabilità di vincere un duello in base alle armate a propria disposizione*

Come abbiamo visto in precedenza, combattendo ogni volta con un dado solo le probabilità di vincere lo scontro sono così ripartite:

- attacco = 41.7
- difesa = 58.3

In questa versione del codice le probabilità di vittoria sono pesate nel tentativo di simulare il sistema di bilanciamento spontaneo che si dovrebbe venire a creare dal momento che chi ha più armate su un dato territorio, di solito, può propendere a usarne più di una sia in un ipotetico attacco che in una difesa. In questo senso i pesi che abbiamo implementato danno un certo vantaggio a chi, tra i due coinvolti nello scontro, si trovi ad avere più armate sul proprio territorio, e lasciano invariata la probabilità di vincere lo scontro nel caso il numero di armate sia lo stesso per entrambi (infatti la costante *cp* in quel caso si annullerebbe poiché moltiplicata per zero).

Nel caso si stia pesando le probabilità che vinca l'attacco, i pesi verranno così calcolati: assumendo $c(i)$ e $c(j)$ strettamente maggiori di zero la formula che restituirà la probabilità pesata in modo adeguato alla situazione è:

- P(Attacco Vince) + $cp \cdot ((c(i) - c(j)) / (c(i) + c(j)))$

  dove:

  - P(Attacco Vince) = 0.417;
  - *cp* è la costante che si è scelto di utilizzare per i pesi (per esempio 0.3)
  - $c(i)$ è il numero di armate presenti sul territorio attaccante;
  - $c(j)$ è il numero di armate presenti sul territorio che si sta difendendo;

Analogamente per la difesa sarà:

- P(Difesa Vince) + $cp \cdot ((c(j) - c(i)) / (c(i) + c(j)))$

  dove:

  - P(Attacco Vince) = 0.583;
  - *cp* è la costante che si è scelto di utilizzare per i pesi (per esempio 0.3)
  - $c(i)$ è il numero di armate presenti sul territorio attaccante;
  - $c(j)$ è il numero di armate presenti sul territorio che si sta difendendo;

Il fatto di poter utilizzare sempre un solo dado potrebbe non permettere a nessun tipo di tattica di emergere in modo significativo, se non a quella che adotta la forza bruta (ovvero la tattica aggressiva). Questo perché l'aver fissato le probabilità di vittoria di uno scontro, di fatto rischia di appiattire la dinamicità della partita intera, riducendola a una questione di quantità di attacchi svolti, mentre durante una partita è possibile che anche chi non attacca molto spesso si trovi nella condizione di potersi difendere con più armate in confronto alle armate avversarie, avendo in questo modo più chance di cavarsela nei singoli scontri. Una situazione di superiorità numerica dovrebbe infatti avvantaggiare in qualche misura chi vi si ritrova, cosa che invece non succede con un unico

dado. Applicando questi pesi alle probabilità di vittoria di uno scontro, ci si aspetterebbe che una tattica come quella aggressiva adottata più sopra non sia più di gran lunga preferibile ad un'altra. In altre parole, in questo modo non dovrebbe esistere una tattica monotematica e così sbilanciata in grado di garantire a un giocatore probabilità di vittoria in media decisamente più alte.

A parità di tattica e assumendo una costante per il peso *cp* = 0.3 osserviamo quindi che:

	Prob ZERO vince	Prob UNO vince
Passiva	0	0
Neutra	0,517	0,481
Aggressiva	0,478	0,513

Tabella 5.2.8: probabilità di vittoria dei giocatori. Esperimento con probabilità pesata e stessa tattica per entrambi i giocatori

L'assenza dei dati relativi alle partite giocate da entrambi i giocatori con tattica passiva è riconducibile alle motivazioni espresse in precedenza: le partite tendono ad allungarsi indefinitamente (diventano infinite) e il model checker segnala un errore.

E se uno dei due giocatori assume solo ed esclusivamente la tattica aggressiva, in questo caso il giocatore Zero:

	tattica AGGRESSIVA	
	Prob ZERO vince	Prob UNO vince
Passiva	0	0
Neutra	0,624	0,376
Aggressiva	0,482	0,511

Tabella 5.2.9: probabilità di vittoria dei giocatori. Esperimento con probabilità pesata e tattica diversificata

In definitiva sembra che i pesi applicati alle probabilità di vittoria degli scontri non vadano a bilanciare l'andamento delle partite in cui almeno un giocatore adotta una strategia monotematica aggressiva. In aggiunta ai pesi allora proviamo a definire un sistema di regole che permetta al giocatore di scegliere quale tattica adottare in base al numero di armate in suo possesso. Chiameremo questa tattica **dinamica**. Al momento di decidere per un nuovo attacco la decisione verrà presa adottando la tattica:

- **Passiva** quando si ha un numero di armate <= 2;
- **Neutra** quando se ne ha esattamente 3;
- **Aggressiva** con le armate >3 ;

Si è proceduto quindi a un confronto con un giocatore che adotta una tattica di volta in volta diversa, ma costante durante tutta la partita. In questo esperimento avremo quindi la seguente situazione: il giocatore Zero adotterà una tattica dinamica mentre il giocatore Uno ne adotterà di volta in volta una tra le tre già utilizzate in precedenza:

	Tattica DINAMICA	
Tattica adottata da UNO	Prob ZERO vince	Prob UNO vince
Passiva	0	0
Neutra	0,657	0,336
Aggressiva	0.4774	0.5233

Tabella 5.2.10: robabilità di vittoria dei giocatori. Esperimento con probabilità pesata e tattica diversificata

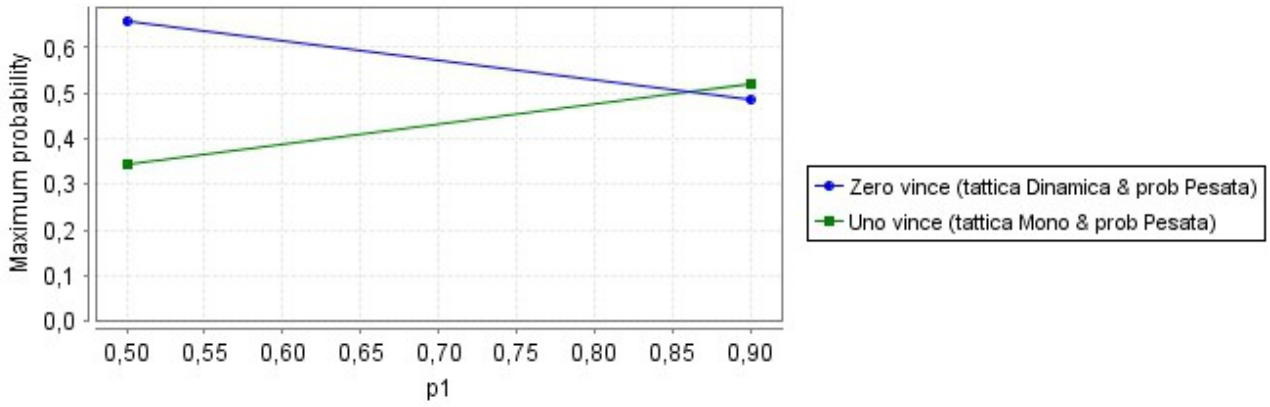


Figura 5.2.8: grafico che riporta i valori mostrati nella tabella precedente

Purtroppo il giocatore Zero, ovvero quello che adotta una tattica dinamica simulando di volta in volta una sorta di scelta basata sullo stato delle sue risorse, non riesce a spuntarla contro la forza bruta di un approccio monotematico e sbilanciato come quello della tattica aggressiva.

Ci siamo chiesti allora cosa succederebbe se il giocatore Zero, per esempio, temporeggiasse per qualche turno attaccando di rado, e successivamente iniziasse ad adottare la tattica aggressiva. Cosa succederebbe quindi se Zero adottasse una tattica in sostanza dinamica che però basa le decisioni sul numero di turni giocati e non sul numero di armate in proprio possesso (questa strategia si propone di chiamarla **attendista**). Quelli che seguono sono i risultati dell'esperimento condotto facendo adottare questa nuova tattica dinamica al giocatore Zero e lasciando ancora una volta al giocatore Uno la tattica aggressiva. Sull'asse delle ascisse all'interno del grafico (e nella prima colonna di sinistra) si trovano i valori relativi al numero di turni giocati dal giocatore Zero adottando la tattica passiva, ossia i turni “attesi”. Si avrà quindi il monitoraggio di quattro diverse modulazioni della tattica attendista.

	ATTENDISTA	AGGRESSIVA
turni di attesa	Prob ZERO vince	Prob UNO vince
10	0.683	0,314
20	0.648	0.349
30	0.629	0.379
40	0.623	0.373

Tabella 5.2.11: robabilità di vittoria dei giocatori. Esperimento con probabilità pesata e tattica attendista per Zero, tattica aggressiva per Uno

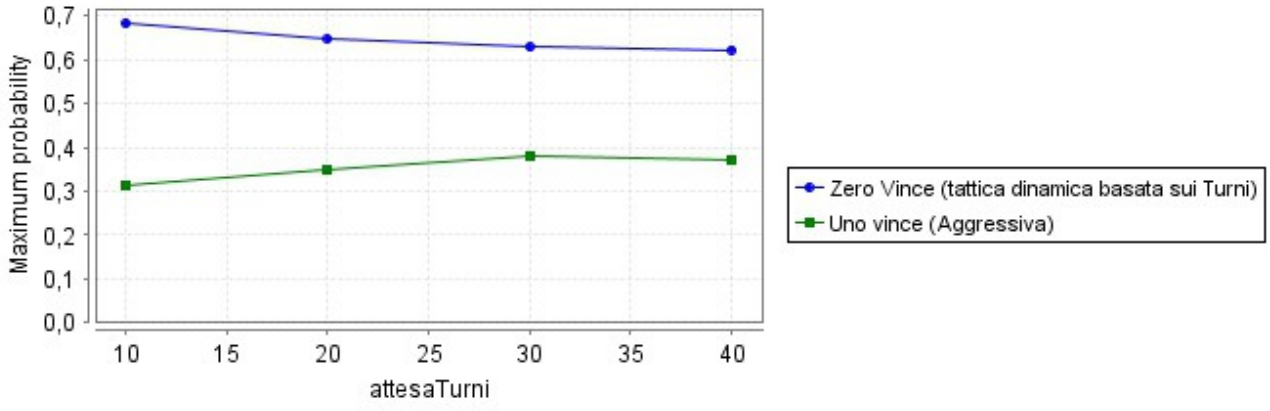


Figura 5.2.9: grafico che riporta i valori mostrati nella tabella precedente

Da questo grafico è possibile evincere che un numero di turni di "attesa" compreso tra 10 e 20 possa dare ottime chance di vittoria al giocatore che adotta questo tipo di strategia, anche se gioca contro un avversario che attacca nove volte su dieci. Attendere molto, di contro, se non produce effetti molto penalizzanti, comunque influisce in modo negativo. L'efficacia di una tale strategia risiede molto probabilmente nel fatto che, partendo da un setup identico per entrambi i giocatori, un maggior numero di attacchi andrà a indebolire chi li sferra dal momento che a parità di armate l'attacco è svantaggiato. Un volta perse un certo numero di armate in questo modo, la conseguenza sarà quella di una maggiore vulnerabilità e la possibilità di soccombere agli attacchi sferrati in maniera incalzante da un avversario in possesso di più armate. Con la seguente tabella si intende mostrare come, nel caso in cui entrambi i giocatori adottino la stessa strategia che basa la propria

dinamicità sul numero di turni giocati, risulti essere più probabile vincere per il giocatore che saprà attendere più a lungo (la tabella relativa alla probabilità che Uno ha di vincere è ricavabile dalla seguente):

Probabilità che vinca ZERO						
		Turni aspettati da giocatore ZERO				
Turni aspettati da giocatore UNO	10	20	30	40	50	60
10	0.567	0.705	0.680	0.662	0.663	0.654
20	0.347	0.572	0.714	0.706	0.686	0,681
30	0.367	0.322	0.557	0.738	0.712	0.712
40	0.378	0.336	0.318	0.552	0,732	0.737
50	0.395	0.353	0.310	0.318	0.548	0.743
60	0.397	0.353	0.327	0.295	0.314	0.539

Tabella 5.2.12: tabella che riporta la probabilità di vittoria di Zero relativamente ai turni attesi (tattica attendista)

Osservando la tabella si può osservare un fenomeno: esiste un vantaggio corrispondente a un'attesa maggiore, il quale si traduce in una probabilità più alta di vincere (si vedano i casi evidenziati in azzurro). Una strategia di questo tipo quindi sembrerebbe mettere in evidenza la bontà di un approccio alla partita che preveda una sorta di fase preliminare, durante la quale il giocatore ha il tempo di studiare l'avversario e possibilmente di posizionare in modo strategico le proprie risorse, per poi poter partire all'attacco.

5.2.9 *Conclusioni*

Nell'ambito di questo caso di studio si è inteso modellare un gioco che, aldilà della sua forte componente di *alea*, si caratterizza per essere un gioco da tavolo di strategia. Dopo aver fornito una tipizzazione che vuol delineare in maniera sintetica alcuni dei possibili approcci alla partita, si è proceduto all'osservazione degli effetti di tali tattiche prima sulla durata media delle partite, poi sulla probabilità di vittoria per entrambi i giocatori. In un gioco la cui durata in termini di tempo è

potenzialmente infinita, si è visto come un atteggiamento aggressivo, ovvero molto propenso agli scontri diretti con l'avversario, comporti inevitabilmente un accorciamento delle partite stesse. Pertanto nel caso in cui entrambi i giocatori adottino tale strategia, è possibile che l'esperienza di gioco derivante possa essere in qualche modo depotenziata dal "prosciugamento" troppo rapido delle risorse: accorciando drasticamente le partite è infatti plausibile che alcune delle dinamiche naturali di un gioco di strategia ad alta interazione possano venir meno.

L'obiettivo dei vari test è stato anche la ricerca di una possibile tattica predominante, la quale effettivamente sembra esistere ma costituisce in sé una specie di paradosso: più precisamente, si può notare come la conseguenza della meccanica che regola le armate bonus sia un progressivo accumulo di armate da parte dei giocatori, i quali potrebbero pertanto scegliere di non attaccare mai con l'obiettivo di mantenere sempre un livello di risorse sufficiente a garantirsi la permanenza in gioco. In questo modo però, se entrambi si trovassero ad adottare una tattica del genere, come si è visto, le partite diventerebbero infinite e pertanto non ci sarebbe nessun vincitore. Sempre in questa stessa situazione, si è poi osservato come "spezzare" una strategia per così dire di attesa sferrando un'offensiva duratura all'avversario possa invece essere molto remunerativo. Nel caso in cui, ancora una volta, questo approccio dovesse venir adottato da entrambi i giocatori, si è potuto constatare che sarà più favorito colui che farà trascorrere più turni prima di iniziare l'offensiva. Sembra non esistere quindi un'unica tattica dominante che sia possibile mettere in pratica, poiché neanche l'approccio di forza bruta della tattica aggressiva può garantire una probabilità di vincere più alta di quella dell'avversario in ogni possibile partita. Ciò sembra tradursi nella realtà in un'inevitabile complicazione delle strategie vincenti, le quali, così almeno ci suggerisce quest'ultimo esperimento, dovranno tenere di conto sia lo stato delle proprie risorse individuali che l'agire dell'avversario, possibilmente sfruttando in modo positivo (limitando le perdite in termini di risorse) quella che potrebbe essere una fase iniziale di studio del nemico.

5.3 Studio di proprietà legate alle scelte dei giocatori in un modello che rappresenta un gioco cooperativo del genere dungeon crawl

Figura 5.3.1: i quattro eroi protagonisti del gioco Heroquest

Il modello è ispirato al famoso gioco da tavolo HeroQuest [28], riferimento al sito in fondo:] ideato da Stephen Baker e pubblicato in Europa nel 1989 dalla Milton Bradley Company. La versione originale ha un'ambientazione fantasy e vede quattro eroi intenti a sconfiggere le forze del male disposte sulla plancia di gioco da un *master*, ovvero un giocatore che "interpreta" la parte del gioco stesso. Il master è chiamato a creare dei *dungeons* (celle sotterranee) utilizzando le miniature a disposizione. All'interno di questi dungeons i giocatori troveranno i vari nemici da sconfiggere con l'ausilio di poteri e armi speciali. Il modello intende simulare quindi un gioco cooperativo nel quale i giocatori devono esplorare un numero predefinito di stanze all'interno delle quali si trovano di volta in volta un numero di entità nemiche compreso tra 4 e 7. Per procedere alla stanza successiva i

giocatori devono sconfiggere tutte le entità nemiche (che d'ora in avanti chiamerò *mostri*) presenti nella stanza in cui si trovano. Una volta che la stanza sarà completamente “disinfestata” dai mostri, i giocatori procederanno tutti insieme ad esplorare la stanza successiva, fino ad arrivare nell'ultima stanza all'interno della quale si troveranno a dover fronteggiare il Mostro Finale (MF).

5.3.1 *Caratteristiche dei giocatori*

I giocatori sono in tutto quattro e sono descritti da altrettante variabili. Ogni giocatore sarà dotato all'inizio di ogni partita di un numero predefinito di punti ferita: il giocatore che perde un combattimento ne consuma uno. Una volta esauriti i punti ferita la variabile relativa allo stato del giocatore prende valore pari a zero e "segnala" al modello che il giocatore in questione è stato eliminato e non può più partecipare al gioco, ovvero verrà saltato il suo turno.

5.3.2 *Caratteristiche dei mostri*

Ogni volta che i giocatori entrano in una nuova stanza questa si riempe istantaneamente di un numero casuale di mostri compreso tra 4 e 7. I mostri sono costituiti da un solo punto ferita; questo significa che perdendo uno scontro vengono eliminati. Caratteristiche del Mostro Finale: il MF è anch'esso costituito da un solo punto ferita ma a differenza degli altri mostri che combattono di volta in volta contro un singolo giocatore, questo combatte contro tutti i giocatori che sono riusciti a sopravvivere e a raggiungere l'ultima stanza.

5.3.3 *Modalità degli scontri*

Esistono due tipi di combattimento:

- combattimento tra un mostro e un singolo giocatore;
- combattimento tra il MF e i giocatori giunti vivi all'ultima stanza;

Per quanto riguarda il primo, il giocatore che si trova a combattere contro un mostro deve tirare due dadi mentre il mostro ne tira uno soltanto. Dopodiché si confronta il valore più alto ottenuto dal giocatore con il punteggio totalizzato dal mostro. Il giocatore vince, e quindi elimina il mostro, se realizza un punteggio maggiore o uguale a quello dell'avversario.
Chi tra i giocatori sopravviverà agli scontri riuscendo ad arrivare fino all'ultima stanza si troverà ad affrontare il MF, il quale lancerà quattro dadi sommando i vari punteggi. Quindi sarà il turno dei giocatori che lanceranno a loro volta due dadi ciascuno. Per ogni giocatore verrà scelto il punteggio più alto da sommare a quello degli altri. Come per gli scontri precedentemente descritti, anche in questo i giocatori vincono totalizzando un punteggio uguale o maggiore a quello del MF.

La partita finisce quando i giocatori sconfiggono il MF, oppure quando nessun giocatore rimane in vita.

5.3.4 *Analisi del modello*

Gli esperimenti condotti qui di seguito prevedono un numero di stanze pari a 10.
Le proprietà saranno simulate e non verificate poiché la grandezza del modello e i limiti di memoria non permettono di svolgere verifiche in tempi ragionevoli. Il sample utilizzato prevede 10000

partite scelte casualmente tra tutte le partite possibili.

Per rendere divertente il gioco senza che il risultato sia troppo prevedibile si assume che una probabilità P = (0,5) di vincere da parte dei giocatori (e dei mostri) possa essere accettabile.
Ma perché si gioca a un gioco collaborativo? Assumendo che il fine ultimo di un giocatore che si cimenta in questo genere di attività ludica sia il divertimento, e che questo divertimento sia dato dal giocare assieme ad altri giocatori, collaborando e tentando di raggiungere un obiettivo collettivo (ma non solo), si può pensare di porsi alcune domande:

- Cosa succede se un giocatore viene eliminato troppo presto dal gioco? Come fare per evitare che ciò accada?
- Cosa può spingere un giocatore a collaborare con un altro?

Il problema dell'uscita precoce dal gioco può contenere due aspetti diversi ma ugualmente importanti: primariamente se un giocatore viene eliminato troppo presto nel corso della partita e la partita dura un tempo considerevolmente lungo il giocatore eliminato non avrà sicuramente una buona esperienza di gioco; in secondo luogo se un giocatore non è tecnicamente eliminato ma è comunque impossibilitato a compiere una qualsiasi azione si corre il rischio che il giocatore in questione sia costretto ad assistere passivamente a gran parte dello svolgimento della partita.
Se si intende tentare di porre rimedio a questi due aspetti dello stesso problema si deve tenere di conto che non basta assicurarsi che i giocatori arrivino il più possibile avanti nello svolgimento della partita, ma che lo possano fare in modo attivo.
Prima di rispondere alla prima di queste due domande è bene procedere indirettamente cercando di sondare lo spazio di possibilità legato all'attività dei giocatori. Quindi potrebbe essere utile chiedersi a che punto della partita viene eliminato il primo giocatore. In un gioco collaborativo la questione

assume una certa rilevanza dal momento che, come si accennava sopra, l'obiettivo dovrebbe essere quello di giocare collaborando con gli altri giocatori. In più si potrebbe osservare che un'eliminazione dal gioco troppo precoce porterebbe un giocatore a restare per tutta la durata della partita in disparte senza la possibilità di poter partecipare al gioco in nessun modo.

Il grafico mostra con quale probabilità i giocatori sono ancora tutti in gioco una volta raggiunta la stanza indicata sull'asse delle ascisse. Le varie curve sono relative al diverso numero di punti ferita con il quale sono stati dotati i giocatori all'inizio della partita (per questi si è deciso di fissare un tetto massimo pari a 10). Più è bassa la probabilità per i giocatori, dato un certo numero di punti ferita iniziali, di rimanere tutti in gioco, più e alta di conseguenza la probabilità di perdere il primo giocatore in una data stanza. La proprietà che si è chiesto al model checker di simulare è:

- P = ? [(sommaG(i) = 4) & (stanzaAttuale = contaStanze)]

dove *sommaG(i)* indica il numero di giocatori attivi ancora in gioco, *stanzaAttuale* è la variabile che rappresenta la stanza in cui si trovano i giocatori e *contaStanze* è una sorta di contatore utilizzato qui allo scopo di effettuare di volta in volta il confronto con la variabile precedentemente descritta. Questa proprietà intende osservare quanto è probabile che la somma dei giocatori sia uguale a 4 negli stati che soddisfano la proprietà (stanzaAttuale = contaStanze) assumendo che, come si può notare sull'asse delle ascisse, *contaStanze* assuma valori interi compresi tra [0,10].

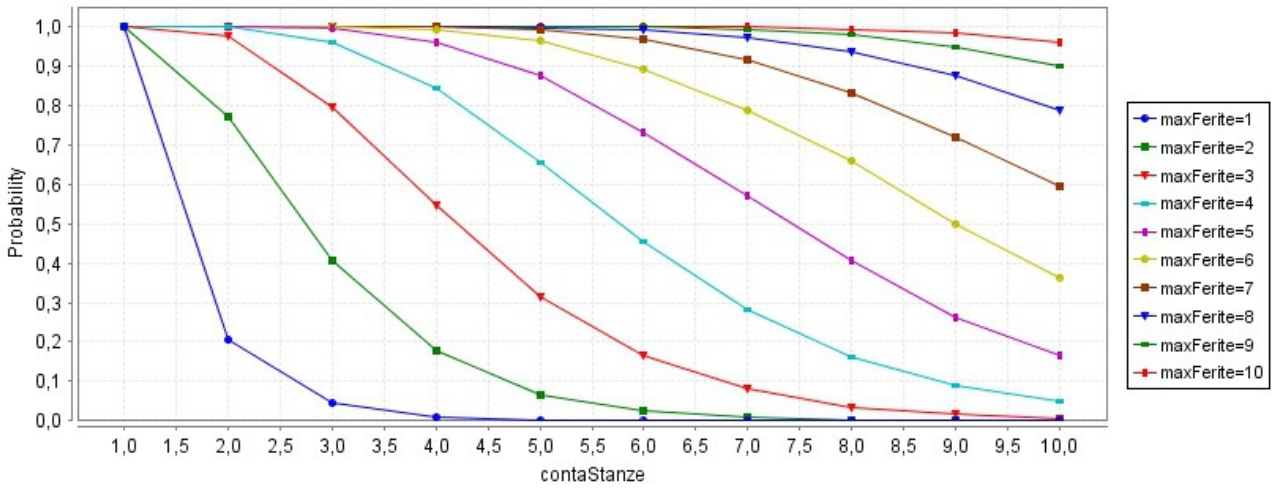


Figura 5.3.2: probabilità di perdere il primo giocatore in corrispondenza di una data stanza (il numero corrispondente ala stanza lo si trova sull'asse delle ascisse)

Si può notare come dotando i giocatori di un numero di punti ferita iniziali pari a 5 o 6 si riesca a garantire che, almeno nella metà dei casi, tutti i giocatori partecipino attivamente a non meno del 75% della partita. Aumentando ancora i punti ferita iniziali aumenterà di pari passo la probabilità che tutti i giocatori rimangano in gioco per tutta la partita, ma se da una parte perdere uno o più giocatori troppo presto è sicuramente un problema, lo è altrettanto l'avere la certezza (o quasi) che nessun giocatore verrà eliminato.

A supporto dell'esperimento di cui sopra, andiamo a vedere nel dettaglio l'andamento della probabilità che tutti i giocatori arrivino all'ultima stanza al variare dell'ammontare dei punti ferita iniziali. Assumendo ancora una volta un tetto massimo per i punti ferita di ogni giocatore pari a 10 si può andare a vedere con quale probabilità i giocatori riescono ad arrivare all'ultima stanza tutti assieme all'aumentare proprio dei punti ferita:

maxFerite	Prob
1	0
2	0
3	0.0057
4	0.0419
5	0.1702
6	0,365
7	0.6003
8	0.7834
9	0.8944
10	0.9558

Tabella 5.3.1: probabilità di arrivare tutti in gioco all'ultima stanza

I risultati dicono che un numero di punti ferita pari a 7 possa essere utile affinché in più della metà dei casi tutti i giocatori raggiungano l'ultima stanza “vivi”.

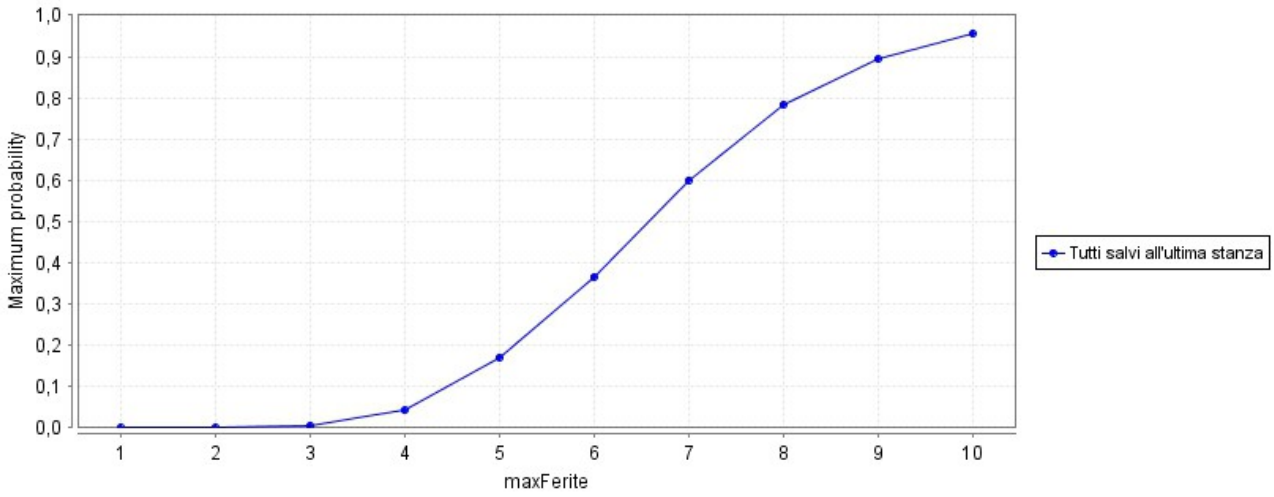


Figura 5.3.3: variare della probabilità che tutti i giocatori arrivino all'ultima stanza ancora in gioco al variare dei punti ferita iniziali

Infine mettiamo a confronto la curva precedentemente ottenuta con le curve relative all'andamento della probabilità che all'ultima stanza arrivino vivi soltanto tre, due e infine un solo giocatore:

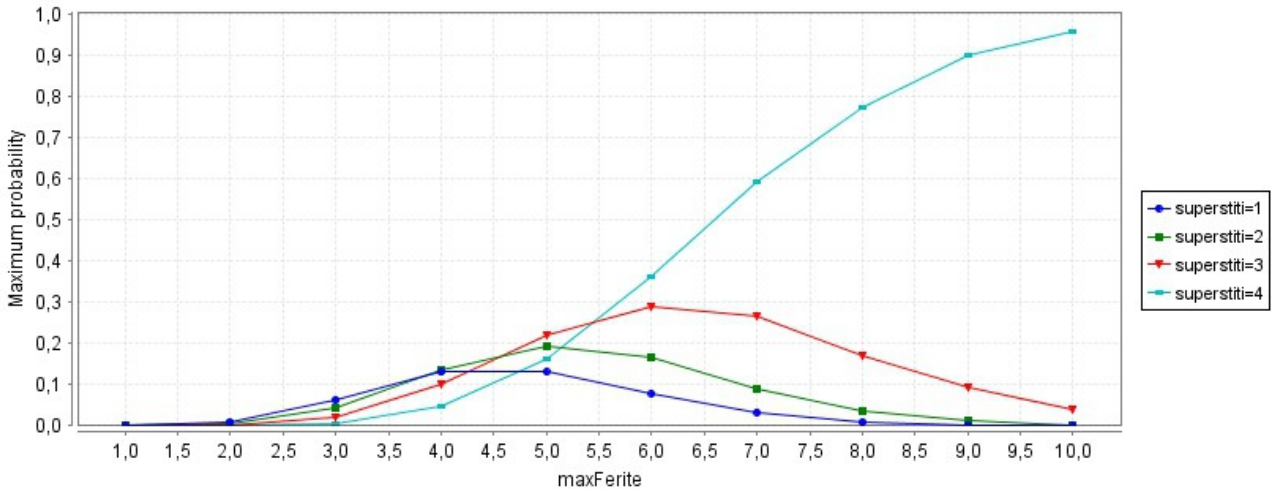


Figura 5.3.4: a seconda del numero dei punti ferita iniziali varia la probabilità che ad arrivare all'ultima stanza ancora in gioco siano 1-4 giocatori (i quattro colori sottolineano i quattro diversi casi)

Tabella contenente soltanto i valori relativi alle partite iniziate con 7 punti ferita per ciascun giocatore:

maxFerite	superstiti	Prob
7	1	0.032
7	2	0.089
7	3	0.267
7	4	0,593

Tabella 5.3.2: probabilità di arrivare in [1-4] giocatori all'ultima stanza partendo con 7 punti ferita

Dunque, fermo restando che lo scopo di un gioco cooperativo sia quello di poter giocare collaborando con gli altri giocatori per perseguire uno scopo comune, il fine ultimo del gioco rappresentato dal modello preso in esame è quello di eliminare tutti i mostri che popolano le varie stanze, compreso il temutissimo MF. Una volta che la probabilità di raggiungere l'ultima stanza tutti insieme si ritiene sufficientemente alta, si può andare a vedere con quale probabilità i giocatori riescano a superare l'ultima prova. Sempre tenendo di conto il numero dei punti ferita a disposizione dei singoli giocatori, andiamo a vedere come varia la loro probabilità di vincere:

maxFerite	Prob
5	0.268
6	0.466
7	0.636
8	0.742

Tabella 5.3.3: probabilità di vittoria dei giocatori

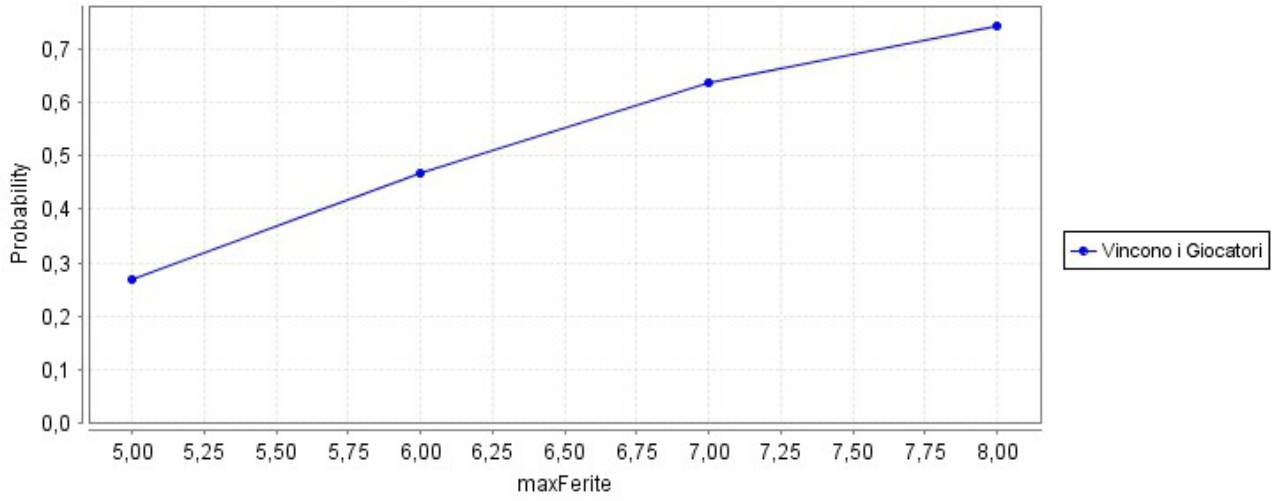


Figura 5.3.5: probabilità che vincano i giocatori in base al numero di punti ferita iniziali

Quest'ultimo grafico mostra in definitiva come un numero massimo di punti ferita pari a 7 possa rappresentare una buona soluzione al fine di "allungare" le partite dei singoli giocatori e di dare loro la possibilità di giocarsi la vittoria contro il MF con la probabilità di sconfiggerlo in più del 50% dei casi.

Riassumendo si può pensare che con un simile "equipaggiamento", con una probabilità più alta di P = (0,5), i giocatori siano in grado di raggiungere lo scopo ultimo del gioco, l'eliminazione di tutti i nemici, e di farlo tutti quanti insieme. Resta però in parte irrisolto il problema dell'uscita precoce dal gioco da parte dei giocatori: nonostante le verifiche fin qui fatte indichino una possibile buona soluzione per intervenire sulla durata della partita e, più in generale, anche sulla probabilità di vittoria, si può pensare di migliorare ancora questo aspetto in modo da minimizzarlo. In altre parole si può tentare di rendere ancora più difficile perdere uno o più gocatori durante il game-play. In

linea di massima esistono due approcci utili per tentare di arginare il rpoblema:

- facilitare le fasi intermedie del gioco;
- tentare di ottenere l'effetto "o tutti o nessuno";

L'effetto a cui si fa riferimento nel secondo caso intende delineare un game-play in cui, data la forte componente cooperativa di un gioco, sia difficile giocare senza uno o più giocatori, al punto che una volta eliminato il primo saranno inevitabilmente eliminati anche tutti gli altri nel giro di pochi turni. Questo permetterà ai giocatori eliminati per primi di non attendere troppo senza poter giocare e, una volta eliminati tutti i giocatori, di iniziare una nuova partita tutti insieme.
Nei prossimi paragrafi si propongono due diverse soluzioni che in qualche misura abbracciano prima uno, poi l'altro approccio a cui si è appena fatto cenno.

5.3.5 *Una possibile meccanica che allunghi le partite dei singoli giocatori: serbatoio di punti ferita condivisi*

Questo esperimento è stato fatto con una versione di codice che prevede un serbatoio di punti ferita condivisi fruibile da tutti i giocatori per un numero finito di volte. Ogni volta che un giocatore attinge al serbatoio recupera tutti i punti ferita che aveva all'inizio della partita.

Cosa succede allora se si da la possibilità ai giocatori di "resuscitare" nel corso della partita? La questione relativa all'uscita precoce dalla partita è da tenere in grande considerazione, pertanto una scelta di design che potrebbe in qualche misura offrire una soluzione a questo genere di problematica (oltre a quella già vista di scegliere con accuratezza il numero di punti ferita iniziali) è quella che offre ai giocatori, una volta esauriti i propri punti ferita, di riacquistarli per un numero predefinito di volte. Una volta esaurite anche le scorte di punti ferita extra il gioco proseguirà normalmente senza che chi viene eliminato abbia la possibilità di partecipare ulteriormente. Meccaniche di questo tipo vedono spesso coinvolti oggetti dai poteri magici in grado di donare una nuova vita, pozioni e filtri medicamentosi oppure semplici kit di pronto soccorso: in ogni caso lo scopo è sempre lo stesso, ossia dare la possibilità al giocatore che si trova a essere eliminato di continuare a giocare. Un esempio di un gioco cooperativo di recente pubblicazione e grande successo che utilizza una meccanica che prevede la cura dei compagni in difficoltà è *Zombicide* [28], nel quale alcuni sopravvissuti tentano tutti insieme di contrastare all'invasione di un'orda di terribili *zombie*.

Da parte nostra la scelta di offrire punti ferita extra è stata fatta in questo modo: viene inserito nel modello una sorta di "serbatoio" di punti ferita extra condiviso e fruibile da ciascuno dei giocatori fino al suo esaurimento. I giocatori possono "ricaricarsi" e rientrare in gioco semplicemente stando fermi un turno (senza combattere con alcun mostro). Ogni giocatore può rifornirsi di punti ferita

extra tante volte quante ne avrà bisogno, l'unico limite sarà la capienza predefinita del serbatoio. Ogni volta che un giocatore “resuscita” riacquista tutti i punti ferita che aveva all'inizio della partita. Resta ora da capire quale combinazione di parametri adottare al fine di non stravolgere il gioco stesso e di mantenerlo comunque divertente da giocare, ovvero di mantenere l'esito sufficientemente incerto. Resta quindi da capire quanti punti ferita iniziali saranno necessari per ottenere un risultato come quello precedentemente ottenuto, ma soprattutto quante “ricariche” inserire nel serbatoio di punti ferita.

Nel grafico che segue si mostra con quale probabilità i giocatori arrivano tutti all'ultima stanza al variare sia dei punti ferita ricevuti all'inizio della partita, sia del numero di ricariche contenute nel serbatoio:

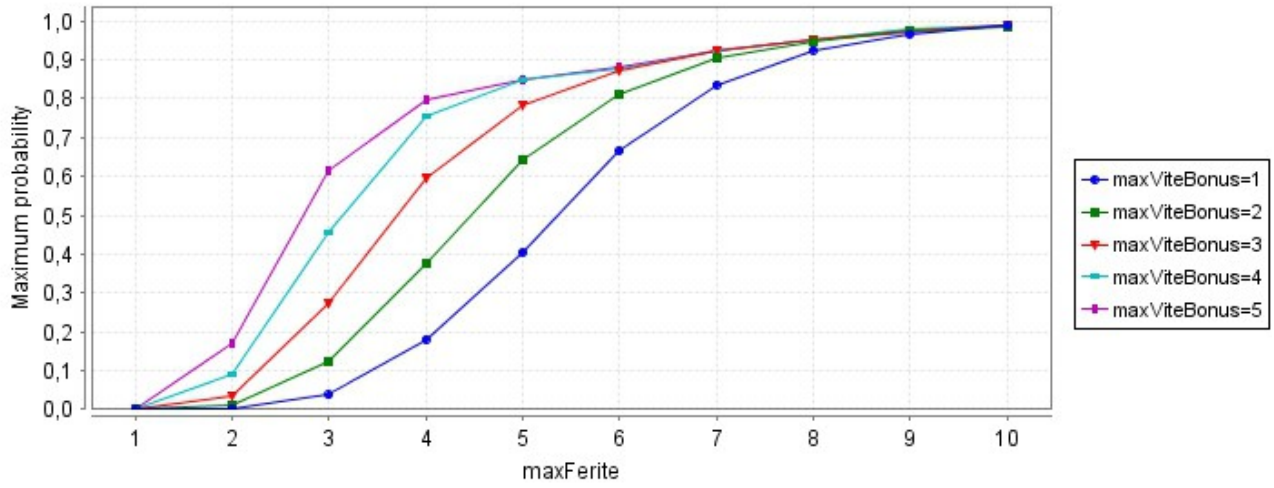


Figura 5.3.6: probabilità che tutti i giocatori arrivino ancora in gioco all'ultima stanza adottando varie combinazioni di punti ferita iniziali e ricariche bonus

Dal grafico si nota come, per rimanere su una soglia di probabilità di circa P = (0,5) esistano varie combinazioni di parametri che in modo analogo offrono una possibile soluzione al designer. Inoltre si nota come il numero di punti ferita necessari ad ogni giocatore affinché tutti possano arrivare

all'ultima stanza insieme si abbassa di qualche unità. Questa conseguenza è con buona probabilità dettata dal fatto che il serbatoio di ricariche costituisce una sorta di moltiplicatore di punti ferita. Ogni giocatore potrà, almeno in potenza, usufruire di molti più punti ferita nel corso di una partita rispetto alla versione del gioco che non prevede questo tipo di bonus. Se si considerano i punti ferita globalmente, si può vedere come dotando i giocatori di 7 punti ferita iniziali, il totale dei punti ferita in gioco con quattro giocatori sarà pari a:

- $(7 \cdot 4) = 28$

Guardando il grafico poco sopra si nota come, in generale, le combinazioni che permettono di ottenere risultati analoghi a quelli ottenuti nell'esperimento svolto in precedenza vadano a produrre in realtà un totale di punti ferita per così dire spendibili dai giocatori, che differisce di poco (se non affatto) dal totale calcolato poco sopra:

- con 5 punti ferita iniziali e 2 “ricariche” bonus → $(5 \cdot 4) + (5 \cdot 2) = 20 + 10 = 30$
- con 6 punti ferita iniziali e 1 “ricariche” bonus → $(6 \cdot 4) + (6 \cdot 1) = 24 + 6 = 30$

In entrambi gli esempi la probabilità che tutti i giocatori raggiungano l'ultima stanza risulta essere comunque compresa tra i valori ottenuti in precedenza dotando ciascun giocatore di 7 o 8 punti ferita iniziali:

maxFerite	maxViteBonus	Result
5	2	0.651
6	1	0.660

Tabella 5.3.4: performance relative a due possibili combinazioni di punti ferita e vite bonus

Si mostra di seguito anche la probabilità che ad arrivare all'ultima stanza siano tre, due e infine un solo giocatore:

Numero di superstiti = 1		
maxFerite	maxViteBonus	Result
5	2	0,018
6	1	0,027

Numero di superstiti = 2		
maxFerite	maxViteBonus	Result
5	2	0,08
6	1	0,083

Numero di superstiti = 3		
maxFerite	maxViteBonus	Result
5	2	0,249
6	1	0,221

Tabella 5.3.5: tabelle da usare come confronto con i risultati mostrati in Tabella 5.3.2

5.3.6 *Cosa spinge un giocatore ad aiutare un suo compagno? La cooperazione*

Se le scelte di implementazione fatte fino a ora (aumentare o diminuire le risorse a disposizione dei giocatori, dare loro la possibilità di tornare in gioco una volta eliminati) possono offrire delle soluzioni più o meno buone per quanto riguarda la problematica dell'uscita precoce dei giocatori dalla partita, lo stesso non si può dire per l'aspetto collaborativo che caratterizza un gioco come questo, nel quale i giocatori agiscono all'unisono affrontando una o più minacce comuni. Di fatto soluzioni del genere non contribuiscono a mettere i giocatori di fronte a scelte di tipo cooperativo (il gioco in questa fase è più simile a una gara tra singoli giocatori). In altre parole in questo modo i giocatori non si trovano a dover scegliere tra un vantaggio personale e uno collettivo, dove il vantaggio collettivo magari comporta una qualche rinuncia individuale. Per dare la possibilità ai giocatori di compiere scelte che vadano in questa direzione si può pensare di inserire un ulteriore elemento: il giocatore che rimane senza punti ferita può essere "curato" dagli altri giocatori che in questo modo gli permettono di continuare a giocare. Appena un giocatore esaurisce i suoi punti ferita, uno degli altri giocatori che ne ha ancora a disposizione può decidere se curarlo o meno; in caso decidesse di farlo dovrà rinunciare a due dei propri punti ferita così da regalarne uno al giocatore rimasto senza. In caso contrario i giocatori ancora in vita continuano a giocare nello stesso modo di prima. Lo scopo di questo meccanismo dovrebbe essere quello di aumentare la probabilità di arrivare all'ultima stanza tutti insieme, a partire da un numero uguale o possibilmente inferiore di punti ferita iniziali, mettendo allo stesso tempo i giocatori di fronte a scelte per così dire cooperative. La scelta di curare o meno un giocatore è legata al parametro p il quale indica la probabilità con la quale tale scelta prenderà una direzione piuttosto che un'altra. Più precisamente p indicherà la probabilità con la quale i giocatori aiuteranno il compagno o i compagni rimasti senza punti ferita, e $(1-p)$ determinerà l'evenienza opposta nella quale il giocatore di turno decide di non

cedere i suoi preziosi punti ferita. In questo primo esperimento la scelta sarà quindi guidata da *p* e dal numero dei giocatori che sono rimasti senza punti ferita. In questo test il parametro *p* prenderà un valore pari a 0,5 facendo sì che ogni volta che uno o più giocatori saranno senza punti ferita il giocatore di turno sceglierà in maniera equiprobabile se aiutare uno dei compagni, oppure se continuare a giocare senza di loro:

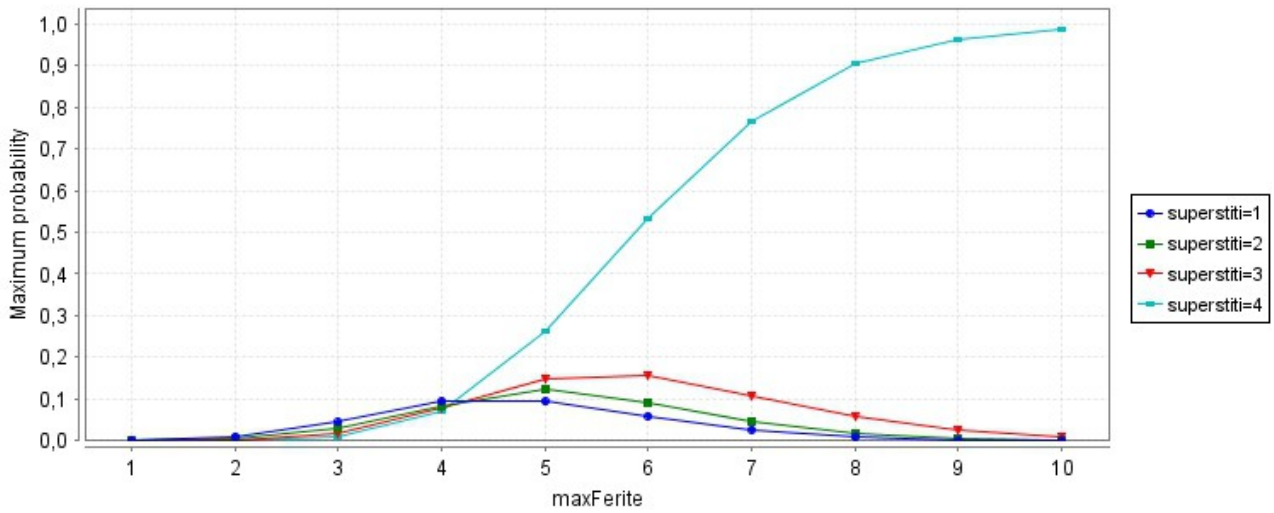


Figura 5.3.7: a seconda del numero dei punti ferita iniziali varia la probabilità che ad arrivare all'ultima stanza ancora in gioco siano 1-4 giocatori (i quattro colori sottolineano i quattro diversi casi)

Di seguito il dettaglio dei record delle partite giocate con 7 punti ferita in partenza (colonna di sinistra). La probabilità nell'ultima colonna di destra è relativa al numero di giocatori rimasti in vita all'ultima stanza indicato nella colonna centrale:

"Meccanica delle cure"		
maxFerite	superstiti	Probabilità
7	1	0.026
7	2	0.044
7	3	0.106
7	4	0.766

Tabella 5.3.6: probabilità di arrivare in [1-4] giocatori partendo con 7 punti ferita

Confrontando questo grafico e i valori riportati in tabella con quelli analoghi in *Figura 5.3.4* e *Tabella 5.3.2* sembra possibile affermare che una delle conseguenze di questa meccanica sia l'aumentare della difficoltà delle singole partite. Più precisamente l'aumentare della difficoltà di giocare quando si perdono uno o più giocatori. In altre parole ancora sembra che, inserendo questa meccanica che di fatto genera una certa perdita gratuita di punti ferita nel corso della partita e considerando partite giocate con uno stesso numero di punti ferita iniziali, sia meno probabile che all'ultima stanza arrivino uno, due o tre giocatori e contemporaneamente sia sensibilmente più probabile che ad arrivarci siano in quattro (la differenza è di |0,173|). Complessivamente diminuisce quindi la probabilità che soltanto qualcuno dei giocatori arrivi all'ultima stanza, ossia "o tutti o nessuno" sembra essere la direzione che prende il gioco adottando una meccanica del genere.

Cosa cambia modificando il valore di *p*? Potrebbe essere interessante vedere se mantenendo un atteggiamento globalmente più o meno generoso la probabilità di raggiungere l'ultima stanza tutti insieme subisce qualche innalzamento o abbassamento considerevole. Nel grafico che segue si mettono a confronto le tre serie di dati: in blu quelle relative a un atteggiamento poco cooperativo, in verde atteggiamento neutro, in rosso i risultati derivanti da partite giocate in modo spiccatamente cooperativo.

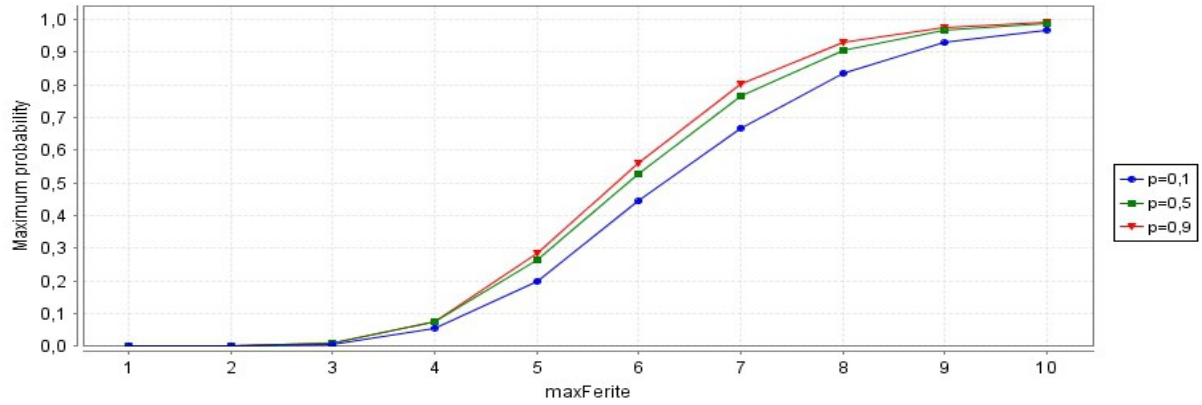


Figura 5.3.8: diverse performance a seconda del grado di cooperazione espresso dai giocatori
blu = poco collaborativo
verde = neutro
rosso = molto collaborativo

Qui sotto si raccolgono soltanto i risultati ottenuti con una dotazione iniziale di 7 punti ferita:

“Meccanica delle cure”		
maxFerite	p	Prob
7	0.1	0.668
7	0.5	0.765
7	0.9	0.803

Tabella 5.3.7: variare della probabilità di arrivare tutti in gioco all'ultima stanza all'aumentare della cooperazione espressa dai giocatori

Un atteggiamento poco altruista sembra portarci indietro alle performance ottenute con il modello che non prevedeva la meccanica delle cure (anche se leggermente migliori). Si nota anche che un atteggiamento decisamente propenso ad aiutare i compagni rimasti senza punti ferita non produce risultati così radicalmente distanti o migliori rispetto all'atteggiamento che fa prendere ai giocatori le decisioni di volta in volta in maniera equiprobabile. La differenza che corre tra i risultati prodotti con partite giocate in modo poco altruista (in blu), e partite giocate da giocatori decisamente più collaborativi (in rosso) resta comunque sensibile (|0,135|).

Quindi si è andati a modificare la strategia con la quale la meccanica delle cure viene messa in moto inserendo una soglia di punti ferita complessivi (ovvero che si riferisce alla somma totale dei punti ferita di tutti i giocatori) sotto la quale i giocatori non aiuteranno i compagni in difficoltà, e sopra la quale invece cederanno i loro punti ferita con probabilità sempre determinata dal valore di *p*. Per questo esperimento i parametri saranno inizializzati nel seguente modo:

- $p = 0.5$;
- maxFerite = 7;
- soglia di punti ferita 5:25:5 (da 5 a 25 per step di 5);

Si nota che con l'innalzarsi della soglia diminuisce la probabilità che tutti i giocatori arrivino all'ultima stanza, e di pari passo aumenta la probabilità che ad arrivarci siano meno di quattro giocatori. Si nota inoltre che se la soglia è fissata a 15 punti o più le probabilità rimangono costanti.

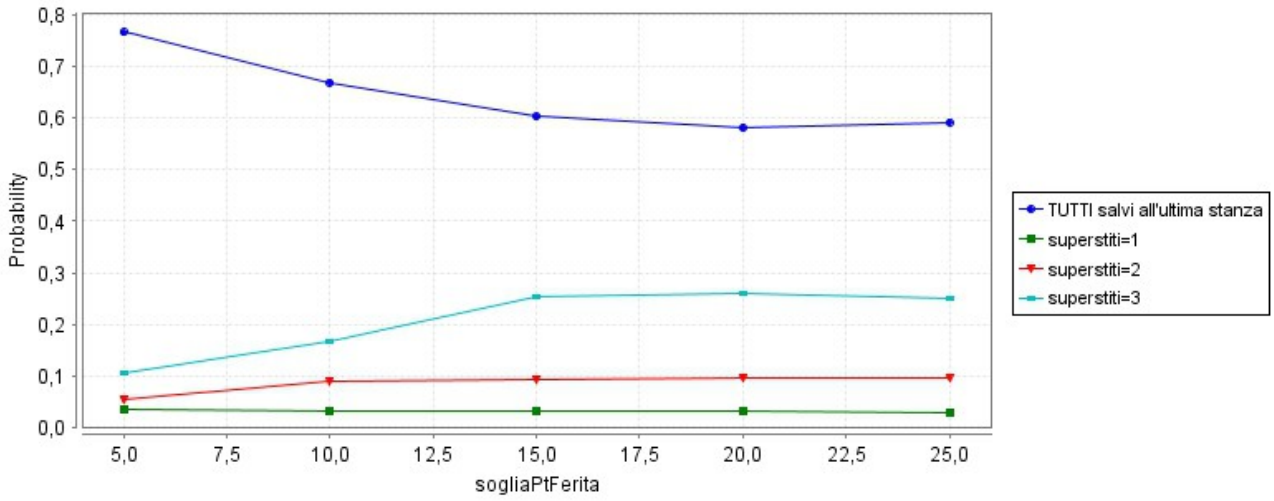


Figura 5.3.9: variare della probabilità di arrivare da [1-4] giocatori all'ultima stanza al variare della soglia di punti ferita oltre sotto la quale non si aiutano i compagni

Nella legenda del grafico con “Tutti salvi” si intende che tutti i quattro giocatori sono arrivati vivi all'ultima stanza. In sostanza sembrerebbe che col diminuire dei casi in cui i giocatori si cedono i punti l'uno con l'altro, diventi meno probabile che tutti quanti arrivino all'ultima stanza per affrontare la sfida finale.

Di seguito il monitoraggio delle tre diverse curve ottenute impostando p prima uguale a 0.1, poi a 0.5 e infine a 0.9 e andando a vedere ancora una volta come varia la probabilità di arrivare tutti salvi all'ultima stanza:

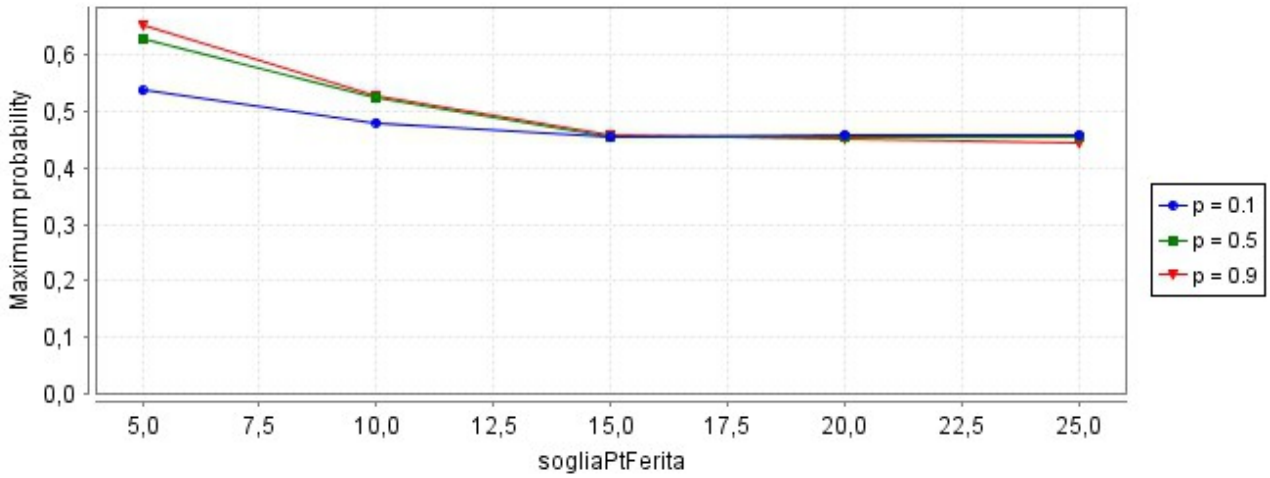


Figura 5.3.10: modulazione della meccanica delle cure con soglia di punti ferita e parametro p

Anche in questo caso non si nota una così ampia differenza tra gli atteggiamenti definiti dal parametro *p,* in special modo se si confronta la curva in rosso con quella in verde. Resta comunque un piccolo scarto tra l'atteggiamento poco altruista (curva blu) e gli altri due.

5.3.7 *Un premio per i giocatori: il bonus finale*

Una possibile scelta per un game designer può essere quella di dare un bonus al giocatore o ai giocatori che compiono una determinata azione o che si ritrovano in una data condizione. Il bonus quindi svolge generalmente la funzione di spingere i giocatori a compiere alcune scelte al posto di altre. Volendo adottare una simile scelta anche per il gioco cooperativo che si sta analizzando, si può pensare di dare la possibilità di lanciare un dado extra qualora ad arrivare all'ultima stanza siano tutti i giocatori. In questo modo al punteggio finale dei giocatori sarà addizionato il risultato del lancio di un ulteriore dado a sei facce. Se questo non può certo influire sulla probabilità che tutti i giocatori arrivino in fondo alla partita conservando almeno un punto ferita (almeno nell'ambito del tipo di testing che si sta qui effettuando), può comunque essere interessante andare a vedere come si modifica la probabilità di vittoria se tutti i giocatori riescono a sopravvivere fino al MF o se invece

non ci riescono. In questo caso scegliere di dare un bonus del genere potrebbe incentivare i giocatori a cooperare per assicurarsi una probabilità maggiore di successo. Con il grafico seguente si intende quindi mostrare come varia la suddetta probabilità di vittoria a parità di punti ferita in dotazione all'inizio della partita. I parametri di questo test sono stati fissati come segue:

- $p = 0.5$;
- maxFerite = 7;
- soglia di punti ferita 15;

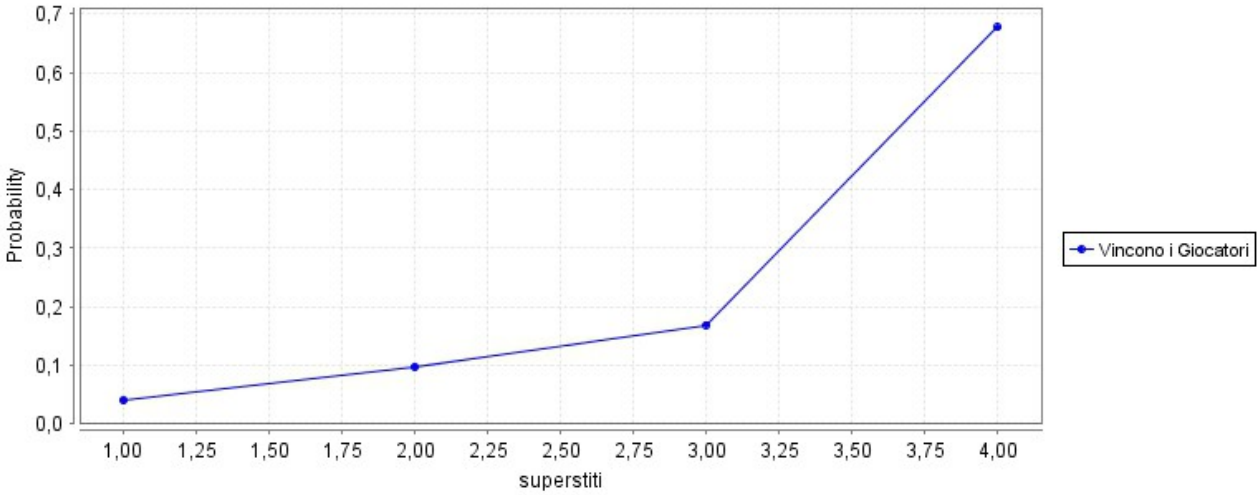


Figura 5.3.11: probabilità di vittoria dei giocatori

superstiti	Prob
1	0.040
2	0.097
3	0.169
4	0.677

Tabella 5.3.8: probabilità di vittoria dei giocatori

Dai risultati pare abbastanza netta la differenza in termini di probabilità di vittoria da parte dei giocatori: in partite in cui la battaglia finale è combattuta da un numero compreso tra uno e tre giocatori la probabilità di vincere per questi non supera lo 0,2 mentre l'ultimo record sottolinea

quanto possa essere d'aiuto combattere tutti insieme il MF.

5.3.8 *Conclusioni*

Alcune delle problematiche legate ai giochi cooperativi hanno una grande rilevanza e sono molto sentite dalla comunità di giocatori, in particolar modo l'eliminazione precoce dalla partita. In questo caso di studio, partendo dall'analisi di un modello di gioco cooperativo, si è tentato di definire un set di parametri (numero di punti ferita, numero di possibili bonus ai giocatori e via dicendo) che fosse in grado di garantire un game-play (esperienza di gioco) di un certo tipo. L'approccio qui utilizzato ancora una volta ha l'obiettivo di sottolineare la possibilità di una metodologia che aiuti il designer a compiere determinate scelte e a indirizzarle dove meglio crede. Infatti i risultati ottenuti sono da considerare soltanto una delle possibili soluzioni ai problemi di volta in volta affrontati.

Le risorse individuali distribuite inizialmente giocano senza dubbio un ruolo fondamentale nello svolgimento delle singole partite: qui si è andati a osservare come varia il numero di giocatori in grado di sopravvivere per tutta la durata del gioco al variare dell'equipaggiamento iniziale nel tentativo di trovare una combinazione adeguata alle possibili aspettative degli stessi giocatori. Come più sopra si è accennato in varie occasioni, può essere infatti di grande aiuto tenere di conto il fattore incertezza: per quanto un gioco estremamente difficile possa non essere poi così attraente, lo stesso si può dire per giochi dal risultato finale troppo prevedibile.

Nel corso dei test si è scelto poi di modificare il modello per cercare di dargli una connotazione più marcatamente cooperativa: andando a inserire una nuova meccanica, ovvero la meccanica soprannominata “delle cure”, si è riscontrato come la cooperazione tra i giocatori sia in grado di influire sulla possibilità che tutti i partecipanti riescano a sopravvivere più a lungo e ad arrivare quindi con più facilità all'ultima stanza. Il parametro p così come la soglia dei punti ferita hanno lo scopo di simulare alcuni dei comportamenti razionali che possibilmente potrebbero essere

riscontrati in un giocatore in carne e ossa. È infatti plausibile che, dato lo stato generale delle risorse disponibili i giocatori decidano o meno di perdere i propri punti ferita in favore dei compagni.

Un ulteriore elemento, il bonus finale, ha reso ancora più centrale il ruolo svolto dalla cooperazione tra giocatori i quali, in questo modo, si trovano di fronte a una scelta che pone da una parte il mantenimento di un vantaggio individuale nei confronti dei compagni che sono stati eliminati (assicurarsi di rimanere in gioco più a lungo) , e dall'altra la possibilità di ottenere a proprie spese un vantaggio collettivo costituito da una maggiore chance di vittoria contro il MF.

6 CONCLUSIONI

In questo elaborato si è cercato di inquadrare l'attività ludica umana partendo dalle definizioni di vari autori; quindi se ne è proposta una nuova che tenga in considerazione diversi aspetti peculiari del gioco. La scelta di concentrarsi su una data tipologia di giochi, i giochi da tavolo, ci ha permesso di circoscrivere il campo di ricerca; ciò è stato utile in quanto ha reso possibile l'individuazione di alcune specifiche proprietà d'interesse le quali sono solitamente oggetto di studio da parte del game designer. Il game design è infatti considerabile come l'insieme delle tecniche e degli studi volti al miglioramento della fase di progettazione degli artefatti ludici: le pratiche che generalmente vengono messe in atto non sempre prevedono l'ausilio di strumenti di calcolo capaci di esplorare in profondità tutte le possibili evoluzioni dei giochi presi in esame. È per questo motivo che si è quindi tentato di applicare alcune tecniche di verifica formale di modelli all'ambito del game design con l'obiettivo di individuare quindi una metodologia in grado di rendere più efficiente il lavoro del progettista.

Nei tre casi di studio proposti si è inteso esplorare le possibilità di un approccio probabilistico e statistico all'osservazione delle dinamiche che scaturiscono dalle meccaniche dei giochi. In particolare nel primo caso di studio si è tentato di definire un numero ragionevole di caselle speciali (di tipo penalizzante) utile a massimizzarne l'effetto. Date le caratteristiche del modello di gioco implementato, i risultati delle verifiche sembrano indicare l'esistenza di un numero preciso di caselle tale da causare un aumento e una stabilizzazione del loro effetto sulle dinamiche del gioco stesso: pertanto è stato possibile stabilire un numero minimo di caselle di questo tipo in grado di rendere il più possibile efficace l'azione penalizzante nei confronti dei giocatori.

Nel secondo caso di studio si è inteso osservare il variare della durata del gioco al variare

dell'approccio dei giocatori alla partita e, successivamente, esplorare l'esistenza di possibili tattiche dominanti . A fronte dei risultati ottenuti è possibile affermare che in un gioco privo di limiti di tempo prestabiliti e ad alta interazione la durata del game-play sia strettamente vincolata all'atteggiamento adottato dai giocatori: specificatamente sia un approccio aggressivo che uno passivo, se adottati in modo nettamente prevalente nel corso di una partita, rischiano di compromettere la qualità dell'esperienza dei giocatori rispettivamente accorciando drasticamente e allungando all'infinito le partite. Si è notato inoltre che sembra non esistere una sola strategia in grado di garantire una vittoria sicura a chi l'adotta: per meglio dire sembra non esserci una strategia vincente che non prenda in considerazione di momento in momento sia lo stato attuale delle proprie risorse, sia la strategia complessivamente adottata dall'avversario.

Infine, con le verifiche illustrate nel terzo e ultimo caso di studio, si è inteso esaminare alcune proprietà che caratterizzano un gioco cooperativo e affrontare uno dei maggiori problemi riscontrabili in giochi di questo genere, ossia l'eliminazione precoce dei giocatori dalla partita. In particolare è stato possibile effettuare un verio e proprio "tuning" dell'ammontare delle risorse distribuite ai giocatori a inizio partita e, successivamente all'inserimento di nuove meccaniche più strettamente correlate alla cooperatività tra giocatori, ripetere l'operazione in modo da non stravolgere il grado di giocabilità del gioco stesso. Si è poi andati a osservare le conseguenze dell'inserimento di un *bonus* sulle probabilità di vittoria dei giocatori: questo ci ha permesso di quantificare i possibili benefici ricavabili da parte dei giocatori, ovvero di accertarsi che il *bonus* stesso funga da incentivo a compiere determinate scelte e non da deterrente.

Le potenzialità offerte da uno strumento come PRISM model cheker [3] hanno permesso di quantificare le conseguenze di alcune possibili scelte di design, evidenziandone le criticità e gli elementi positivi, in modo da fornire una base concreta

sulla quale apportare eventuali modifiche al gioco che si sta progettando. Uno dei problemi riscontrati durante il lavoro di sperimentazione svolto è la grandezza eccessiva dei modelli implementati, la quale ha reso necessario in alcuni casi il ricorrere a verifiche di tipo statistico effettuate comunque su un numero molto ampio di possibili esecuzioni. I risultati ottenuti vanno comunque nella direzione sperata: la verifica formale può effettivamente restituire in modo molto efficente i limiti del campo di possibilità di un sistema transizionale come quelli visti nei casi di studio, e allo stesso tempo facilita l'esplorazione di nuove soluzioni che possibilmente modificano questo stesso campo. L'approccio che si è inteso sperimentare non è comunque da intendersi in alternatva ai metodi di testing in uso al momento: la componente umana resta imprescindibile per una comprensione profonda del game-play. Ciononostante è possibile affremare che le tecniche di model checking, se applicate allo studio dei giochi, possono offrire prospettive altrettanto feconde e producenti. Una proposta per un approfondimento futuro è la realizzazione di un ambiente di sviluppo nel quale sia possibile costruire in modo semplice modelli di nuovi giochi, verificandone le proprietà e correggendone i difetti durante la stessa fase di implementazione.

7 BIBLIOGRAFIA

[1] E. M. Clarke Jr, O. Grumberg, D. Peled. Model checking. MIT press. 1999.

[2] M. Bertolo, I. Mariani. Game design. Gioco e giocare tra teoria e progetto. Pearson; 2014.

[3] Marta Kwiatkowska, Gethin Norman and David Parker. PRISM 4.0: Verification of Probabilistic Real-time Systems. In Proc. 23rd International Conference on Computer Aided Verification (CAV'11), volume 6806 of LNCS, pages 585-591, Springer. 2011.

[4] Editrice Giochi S.r.l. Sito italiano ufficiale di Risiko [on line]. Disponibile a: http://www.risiko.it/index.php; [consultato il 29 gennaio 2015]

[5] R. Caillois, I giochi e gli uomini. La maschera e la vertigine. V edizione. Milano: Bompiani; 2013. [pag.: 26, 25, 26, 55, 71, 221, 19, 109]

[6] B. Suits. The Grasshopper: Games, life and utopia. Peterborough: Broadview Press; [1978] 2005.

[7] J. Huizinga. Homo ludens. Haarlem: Tijeenk Willing; 1938 (trad. it. Homo Ludens, Torino, Einaudi, 2002) [pag. 13]

[8] K. Salen, E. Zimmerman. Rules of play: game design fundamentals. Cambridge/London: MIT Press; 2004. [pag. 81] [la traduzione è di Bertolo, Mariani]

[9] ©2004-2013 Blizzard Entertainment, Inc. Sito ufficiale italiano che ospita le partite di World Of Warcraft; Disponibile a: https://eu.battle.net/account/creation/wow/signup/; [consultato il 27 gennaio 2015]

[10] J. Nash. Equilibrium points in n-person games. In Proceedings of the national academy of sciences, 36 (1), 1950b.

[11] B.Faidutti; 02/2005; Themes & mechanics 1.0 [on line]; Disponibile a: http://www.thegamesjournal.com/articles/ThemesMechanics1.shtml; [consultato il 27 gennaio 2015]

[12] L.Pulsipher; 02/2006; Another Attempt at Definition [on line]; Disponibile a: http://www.thegamesjournal.com/articles/Essence.shtml; [consultato il 27 gennaio 2015]

[13] Ass. culturale - "TdG" La Tana dei Goblin. Sito per appassionati di boardgames; Disponibile a: http://; www.goblins.net/recensione/carcassonne; [consultato il 27 gennaio 2015]

[14] Ass. culturale - "TdG" La Tana dei Goblin. Sito per appassionati di boardgames; Disponibile a: http://; www.goblins.net/recensione/i-coloni-di-catan; [consultato il 27 gennaio 2015]

[15] Ass. culturale - "TdG" La Tana dei Goblin. Sito per appassionati di boardgames; Disponibile a: http://; www.goblins.net/recensione/caylus; [consultato il 27 gennaio 2015]

[16] Ass. culturale - "TdG" La Tana dei Goblin. Sito per appassionati di boardgames; Disponibile a: http://; www.goblins.net/recensione/hotels; [consultato il 27 gennaio 2015]

[17] S. Appelcline; 09/06/2011; The dice game of stefan Feld [on line]; Disponibile a: http://; www.mechanics-and-meeples.com/2011/06/09/96/; [consultato il 27 gennaio 2015]

[18] C. Strain; 18/08/2014; Random placement in design [on line]; Disponibile a: http://; www.leagueofgamemakers.com/random-placement-in-design/; [consultato il 28 gennaio 2015]

[19] C.Fabricatore; 10/2007; https://; Game play and game mechanics design: a key to quality in videogames [on line]; Disponibile a: www.researchgate.net/publication/236168267_Gameplay_and_game_mechanics_design_a_key_to_quality_in_videogames; [consultato il 28 gennaio 2015]

[20] M. Kwiatkowska, D. Parker. Advances in Probabilistic Model Checking. In Software Safety and Security - Tools for Analysis and Verification, volume 33 of NATO Science for Peace and Security Series - D: Information and Communication Security. Pag. 126-151, IOS Press. June 2012.

[21] E. M. Clarke, W. Klieber, M.ˇs Nov´aˇcek, and P. Zuliani. Model checking and state explosion problem. In Tools for practical software verification. Volume 6659 of LNCS. Pag. 1-30, Springer. 2012.

[22] M. Y. Vardi. Model checking as a reachability problem. In 3rd International Workshop, RP 2009, Palaiseau, France, September 23-25, 2009. Proceedings Volume 5797 of LNCS. Pag 35, Springer. 2009.

[23] PRISM Model Checker. The dining philosopher problem [on line]. In PRISM Tutorial. Disponibile da: http://www.prismmodelchecker.org/tutorial/phil.php; [consultato il 28 gennaio 2015]

[24] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In M. Bernardo and J. Hillston (editors), Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07), volume 4486 of LNCS (Tutorial Volume). Pages 220-270, Springer. June 2007.

[25] J. Hölzl, T. Nipkow. Verifying pCTL model checking. In Tools and Algorithms for the Construction and Analysis of Systems Volume 7214 of LNCS. Pages 347-361, Springer. 2012.

[26] PRISM Model Checker. PRISM Manual version 4.2 [on line]. Disponibile a: http://www.prismmodelchecker.org/manual/ [consultato il 28 gennaio 2015]

[27] Dal forum italiano ufficiale di Heroquest [on line]. Disponibile a: http://www.heroquestgame.com/ThePortal.htm Tutti i diritti relativi al gioco sono di Hasbro - Games Workshop Inc. [consultato il 29 gennaio 2015]

[28] J. "Thorn" Monvoisin, Guillotine Games 2011-2014; dal sito italiano ufficiale di Zombicide [on line]. Disponibile a: http://zombicide.com/it/ [cosultato il 4 febbraio 2015]

8 APPENDICE A

```
//GIOCO DELL'OCA

dtmc


//un giocatore arriva sulla casella finale e la partita finisce
formula fine = ( x + d1 = casellaFinale );

//quando un giocatore lancia il dado e muovendosi finisce oltre la
//casella finale ci rimbalza e torna indietro
formula esubero = (x + d1 > casellaFinale) & (d1 != 0);


//con questo parametro si allunga il tabellone a piacimento
const int casellaFinale;

//partita = 0 -> partita in corso | partita = 1 -> partita finita
global partita : [0..1] init 0;



//MODULO CHE GESTISCE IL TURNO
module Mt


turno : [1..2] init 1;

[turno1] partita = 0  & turno = 1 & d2 = 0-> (turno' = 2);

[turno2] partita = 0 & turno = 2 & d1 = 0 -> (turno' = 1);

endmodule


//MODULO CHE SIMULA LA PARTITA
module M1

//var dado
d1 : [0..6] init 0;

//var giocatore
 x : [0..casellaFinale] init 0;

//azione sincronizzata con il modulo Mt
[turno1] partita = 0 & d1 = 0 ->  1/6 : (d1' = 1) +
                                  1/6 : (d1' = 2) +
                                  1/6 : (d1' = 3) +
```

```
                                        1/6 : (d1' = 4) +
                                        1/6 : (d1' = 5) +
                                        1/6 : (d1' = 6);

[] partita = 0 & d1 != 0 & !esubero & !fine  ->  (x' = x + d1) &
                                                  (d1' = 0);

[] partita = 0 & d1 != 0 & esubero ->  (x' = (casellaFinale - (d1-
                                       (casellaFinale - x) ) ) )
                                       & (d1' = 0);

[] partita = 0  & d1 != 0 & fine  ->   (partita' = 1) &
                                       (x' = casellaFinale);
```

endmodule

```
//sincronizzazione dei moduli. sostituizione letterale delle
variabili come descritto nel costrutto qua sotto
module M2 = M1 [x = y ,  d1 = d2   , turno1 = turno2] endmodule
```

```
//I moduli sono sincronizzati solo sulle azioni che compaiono
//in entranbi, in questo caso la prima del modulo
//M1 e di volta in volta una delle due contenute nel modulo Mt.
//Quando si usa questo costrutto non si può utilizzare il
//simulatore
system Mt || M1 || M2 endsystem
```

GIOCO DI STRATEGIA

dtmc

```
//FORMULE PER L'AVANZAMENTO DEL GIOCO:

//FASE ZERO =  inizializzazione, lettura delle variabili
//             territorio e carrarmatini;

//FASE UNO =   passaggio sopra i territori per capire qualisono
//             disponibili per l'attacco (da parte del giocatore di
//             turno);

//FASE DUE =   scelta del territorio che si dovrà difendere e di
//             quello che dovrà attaccare

//FASE TRE =   si svolge la battaglia

//FASE QUATTRO =    eventuale spostamento di armate da un //
//                  territorio ad un altro (che è stato appena
//                  conquistato), quindi il giocatore decide di
//                  attaccare nuovamente o di passare il turno

//FASE CINQUE = partita terminata

formula faseZero = (partita = 0);

formula faseUno = (partita = 1);

formula faseDue = (partita = 2);

formula faseTre = (partita = 3);

formula faseQuattro = (partita = 4);

formula faseCinque = (partita = 5);

formula cambio_giocatore = mod(g +1,2);

//i territori sono in mano a un solo giocatore
formula partitaFinita =  ( (s1 +  s2 + s3 + s4 + s5 + s6) = 0 ) |
                         ( (s1 +  s2 + s3 + s4 + s5 + s6) = 6 );


//labels per stabilire il vincitore
label "ZEROvince" =  ( (s1 +  s2 + s3 + s4 + s5 + s6) = 0 );

label "UNOvince" =  ( (s1 +  s2 + s3 + s4 + s5 + s6) = 6 );
```

```
//formule per rinforzi:

//rinforzi per g = 0:

formula sommaTerrDiZero = ( (1- s1) + (1 - s2) + (1 - s3) + (1 -
s4) + (1 - s5) + (1 - s6) ) ;

formula rinforziZero = floor( sommaTerrDiZero / 2 );

formula rinforziZeroS1 = (1 - s1) * (1/sommaTerrDiZero);
formula rinforziZeroS2 = (1 - s2) * (1/sommaTerrDiZero);
formula rinforziZeroS3 = (1 - s3) * (1/sommaTerrDiZero);
formula rinforziZeroS4 = (1 - s4) * (1/sommaTerrDiZero);
formula rinforziZeroS5 = (1 - s5) * (1/sommaTerrDiZero);
formula rinforziZeroS6 = (1 - s6) * (1/sommaTerrDiZero);

//rinforzi per g = 1:

formula sommaTerrDiUno = (s1 + s2 + s3 + s4 + s5 + s6);

formula rinforziUno = floor( ( sommaTerrDiUno) / 2 );

formula rinforziUnoS1 =  s1 * (1/sommaTerrDiUno);
formula rinforziUnoS2 =  s2 * (1/sommaTerrDiUno);
formula rinforziUnoS3 =  s3 * (1/sommaTerrDiUno);
formula rinforziUnoS4 =  s4 * (1/sommaTerrDiUno);
formula rinforziUnoS5 =  s5 * (1/sommaTerrDiUno);
formula rinforziUnoS6 =  s6 * (1/sommaTerrDiUno);
//formule delle guardie

formula guardiaTerritorioConquistato = (c1 = 0 | c2 = 0 | c3 = 0 |
                                        c4 = 0 | c5 = 0 | c6 = 0);


formula guardiaDif = (dif = 0);



//FORMULE PER ATTACCABILITA' G = 0:
//s1 attaccabile da g = 0
formula AttaccabileS1DaGzero =     s1 * (floor((cs2 + cs3)/2) +
                                   mod(cs2 + cs3,2));

//s2 attaccabile da g = 0
formula AttaccabileS2DaGzero =     s2 * (floor(((( cs1 + cs5 )/2)+
                                   mod(cs1 + cs5,2)) + cs4) /2) +
                                   mod(((floor((cs1 + cs5)/2) +
                                   mod(cs1 + cs5,2)) + cs4),2));

//s3 attaccabile da g = 0
formula AttaccabileS3DaGzero =     s3 * (floor(((( cs1 + cs5 )/2)+
```

```
                                    mod(cs1 + cs5,2)) + cs4)/2) +
                                    mod(((floor((cs1 + cs5)/2) +
                                    mod(cs1 + cs5,2)) + cs4), 2));

//s4 attaccabile da g = 0
formula AttaccabileS4DaGzero =      s4 * ( floor(((((  cs2 +cs3 )/2)+
                                    mod(cs2 + cs3,2)) + cs6)/2) +
                                    mod(((floor(( cs2 + cs3 )/2) +
                                    mod(cs2 + cs3,2)) + cs6), 2)) ;

//s5 attaccabile da g = 0
formula AttaccabileS5DaGzero =      s5 * (floor(((((  cs2 + cs3 )/2)+
                                    mod(cs2 + cs3,2)) + cs6)/2 ) +
                                    mod(((floor(( cs2 + cs3 )/2) +
                                    mod(cs2 + cs3,2)) + cs6), 2)) ;

//s6 attaccabile da g = 0
formula AttaccabileS6DaGzero =      s6 * (floor(( cs5 + cs4 )/2)+
                                    mod(cs5 + cs4,2));

//somma dei territori attaccabili da g = 0, che dovrebbe fornire
il denominatore

formula sommaAttDaGzero =      (AttaccabileS1DaGzero +
                                AttaccabileS2DaGzero +
                                AttaccabileS3DaGzero +
                                AttaccabileS4DaGzero +
                                AttaccabileS5DaGzero +
                                AttaccabileS6DaGzero );


//FORMULE PER ATTACCABILITA' G = 1:
//s1 attaccabile da g = 1
formula AttaccabileS1DaGuno =       (1-s1) * (floor (( cs2+cs3 )/2)+
                                    mod(cs2 + cs3,2));

//s2 attaccabile da g = 1
formula AttaccabileS2DaGuno =       (1-s2) * floor(((((cs1+cs5 )/2)+
                                    mod(cs1 + cs5,2)) + cs4)/2) +
                                    mod(((floor(( cs1 + cs5 )/2) +
                                    mod(cs1 + cs5,2)) + cs4),2));

//s3 attaccabile da g = 1
formula AttaccabileS3DaGuno =       (1-s3)* (floor(((((cs1+cs5 )/2)+
                                    mod(cs1 + cs5,2)) + cs4)/2) +
                                    mod(((floor(( cs1 + cs5 )/2) +
                                    mod(cs1 + cs5,2)) + cs4), 2));

//s4 attaccabile da g = 1
formula AttaccabileS4DaGuno =       (1-s4) * (floor(((((cs2+cs3)/2)+
                                    mod(cs2 + cs3,2)) + cs6)/2) +
```

```
                                    mod(((floor(( cs2 + cs3 )/2) +
                                    mod(cs2 + cs3,2)) + cs6), 2)) ;

//s5 attaccabile da g = 1
formula AttaccabileS5DaGuno =      (1-s5) * (floor((((cs2+cs3)/2)+
                                    mod(cs2 + cs3,2)) + cs6)/2) +
                                    mod(((floor(( cs2 + cs3 )/2) +
                                    mod(cs2 + cs3,2)) + cs6), 2));

//s6 attaccabile da g = 1
formula AttaccabileS6DaGuno =      (1-s6) * (floor((cs5 +cs4)/2)+
                                    mod(cs5 + cs4,2));


//somma dei territori attaccabili da g = 1, che dovrebbe fornire
il denominatore
formula sommaAttDaGuno =      (AttaccabileS1DaGuno +
                               AttaccabileS2DaGuno +
                               AttaccabileS3DaGuno +
                               AttaccabileS4DaGuno +
                               AttaccabileS5DaGuno +
                               AttaccabileS6DaGuno );



//territori: se vale 0 appartiene al g 0, al giocatore 1
altrimenti

global s1 : [-1..1] init 0;
global s2 : [-1..1] init 0;
global s3 : [-1..1] init 0;
global s4 : [-1..1] init 1;
global s5 : [-1..1] init 1;
global s6 : [-1..1] init 1;


//secondo il regolamento classico ogni giocatore riceve ogni turno
tanti carrarmatini quanti sono i territori da lui occupati, diviso
per 3.
// In questo caso sarebbe 3/3=1
//carrarmatini relativi a ciascun territorio

global c1 : [0..10] init 2;
global c2 : [0..10] init 2;
global c3 : [0..10] init 2;
global c4 : [0..10] init 2;
global c5 : [0..10] init 2;
global c6 : [0..10] init 2;


//queste variabili dicono se è possibile per il giocatore di turno
```

```
attaccare dal territorio s(i)
//zero =  il terr. non è occupato dalle mie armate oppure non ho
abbastanza carrarmatini su quel terr.: NON posso attaccare da quel
territorio
//uno = il territorio è occupato dalle mie armate e ci sono almeno
2 carrarmatini: POSSO attaccare da quel territorio

global cs1 : [-1..1] init -1;
global cs2 : [-1..1] init -1;
global cs3 : [-1..1] init -1;
global cs4 : [-1..1] init -1;
global cs5 : [-1..1] init -1;
global cs6 : [-1..1] init -1;


//costante che stabilisce il numero max di carrarmatini che un
giocatore può avere per ogni territorio
const int carr;


//MODULO CHE SIMULA LA PARTITA
module Mi


g : [0..1] init 0;

partita : [0..5] init 0;

//var contatore
contatore : [0..7] init 0;

//varriabili che memorizzano la scelta del territorio da
attaccare, e quindi si dovrà DIFendere
dif : [0..6] init 0;


//var che memorizzano la scelta del territorio da cui verrà
sferrato l'attacco
att : [0..6] init 0;



//FASEZERO (INIZIALIZZAZIONE E RINFORZI)
//la quantità dei rinforzi è data dalla formula floor(somma dei
territori occupati(g)/3)
//In questo caso, ci sono solo 6 territori perciò i giocatori
riceveranno sempre 1 solo carrarmatino per turno, a meno che non
abbiano
//occupato tutti i territori, ma in quel caso la partita è
terminata.
//QUINDI propongo di usare 2 al denominatore
```

```
//g = 0 riceve i suoi rinforzi e inizia il turno
[]   faseZero & g = 0 &
     contatore = 0->rinforziZeroS1:(c1'=min(c1+rinforziZero,carr))&
                   (partita' = 1) &
                   (contatore' = contatore +1) +

                   rinforziZeroS2:(c2'=min(c2+rinforziZero,carr))
                   & (partita' = 1) &
                   (contatore' = contatore +1) +

                   rinforziZeroS3:(c3'=min(c3+rinforziZero,carr))&
                   (partita' = 1) &
                   (contatore' = contatore +1) +

                   rinforziZeroS4:(c4'=min(c4+rinforziZero,carr))
                   & (partita' = 1) &
                   (contatore' = contatore +1) +

                   rinforziZeroS5:(c5'=min(c5+rinforziZero,carr))&
                   (partita' = 1) &
                   (contatore' = contatore +1) +

                   rinforziZeroS6:(c6'=min(c6+rinforziZero,carr))
                   & (partita' = 1) &
                   (contatore' = contatore +1) ;

//g = 1 riceve i suoi rinforzi e inizia il turno
[]   faseZero & g = 1 &
     contatore = 0->rinforziUnoS1:(c1'=min(c1+rinforziUno,carr)) &
                   (partita' = 1) &
                   (contatore' = contatore +1) +

                   rinforziUnoS2:(c2'=min(c2+rinforziUno,carr)) &
                   (partita' = 1) &
                   (contatore' = contatore +1) +

                   rinforziUnoS3:(c3'=min(c3+rinforziUno,carr)) &
                   (partita' = 1) &
                   (contatore' = contatore +1) +

                   rinforziUnoS4:(c4'=min(c4+rinforziUno,carr))&
                   (partita' = 1) &
                   (contatore' = contatore +1) +

                   rinforziUnoS5:(c5'=min(c5 + rinforziUno,carr))
                   & (partita' = 1) &
                   (contatore' = contatore +1) +

                   rinforziUnoS6:(c6'=min(c6 + rinforziUno,carr))
                   & (partita' = 1)
                   & (contatore' = contatore +1) ;
```

```
//RICAVO CON UN PASSAGGIO SOPRA TUTTI I TERRITORI L'INFORMAZIONE
NECESSARIA.
//QUINDI AVREMO 1 SE IL TERRITORIO E' DISPONIBILE PER L'ATTACCO
DEL GIOCATORE g, E 0 ALTRIMENTI.

//s1
[]   faseUno  & contatore = 1  &
     s1 = g & c1  > 1 -> (cs1' = 1) & (contatore' = contatore + 1);

[]   faseUno  & contatore = 1  &
     ( (s1 = g & c1 <= 1) |
     (s1 = cambio_giocatore) ) ->  (cs1' = 0) &
                                   (contatore' = contatore + 1);

//s2
[]   faseUno  & contatore = 2 &
     s2 = g & c2  > 1 -> (cs2' = 1) & (contatore' = contatore + 1);

[]   faseUno  & contatore = 2 &
     ( (s2 = g &  c2 <= 1) |
     (s2 = cambio_giocatore) ) ->  (cs2' = 0) &
                                   (contatore' = contatore + 1);

//s3
[]   faseUno & contatore = 3 &
     s3 = g & c3  > 1 -> (cs3' = 1) & (contatore' = contatore + 1);

[]   faseUno & contatore = 3 &
     ( (s3 = g & c3 <= 1) |
     (s3 = cambio_giocatore) ) ->  (cs3' = 0) &
                                   (contatore' = contatore + 1);

//s4
[]   faseUno & contatore = 4 &
     s4 = g & c4  > 1 -> (cs4' = 1) & (contatore' = contatore + 1);

[]   faseUno & contatore = 4 &
     ( (s4 = g & c4 <= 1) |
     (s4 = cambio_giocatore) ) ->  (cs4' = 0) &
                                   (contatore' = contatore + 1);

//s5
[]   faseUno & contatore = 5 &
     s5 = g & c5  > 1 -> (cs5' = 1) & (contatore' = contatore + 1);

[]   faseUno & contatore = 5 &
     ( (s5 = g & c5 <= 1) |
     (s5 =     cambio_giocatore) ) ->   (cs5' = 0) &
                                        (contatore' =contatore +1);
```

```
//s6
[]   faseUno & contatore = 6 &
     s6 = g & c6  > 1 -> (cs6' = 1) & (contatore' = contatore + 1);

[]   faseUno & contatore = 6 &
     ( (s6 = g & c6 <= 1) |
     (s6 = cambio_giocatore) ) ->  (cs6' = 0) &
                                   (contatore' = contatore + 1);


[]   faseUno & contatore = 7 -> (partita' = 2);


//FASEDUE (SCELTA DEL TERRITORIO CHE SI DOVRA' DIFENDERE E DI
QUELLO CHE LO ATTACCHERA')
//la formula definisce la probabilità con la quale si attacca un
territorio o un altro, quindi stabilisce quale sarà il territorio
che si dovrà difendere
[] faseDue  & guardiaDif & g = 0 & sommaAttDaGzero > 0 ->

     AttaccabileS1DaGzero/sommaAttDaGzero : (dif' = 1)+


     AttaccabileS2DaGzero/sommaAttDaGzero : (dif' = 2)+


     AttaccabileS3DaGzero/sommaAttDaGzero : (dif' = 3)+


     AttaccabileS4DaGzero/sommaAttDaGzero : (dif' = 4)+


     AttaccabileS5DaGzero/sommaAttDaGzero : (dif' = 5)+


     AttaccabileS6DaGzero/sommaAttDaGzero : (dif' = 6);


[]   faseDue  & guardiaDif & g = 0 &
     sommaAttDaGzero = 0 ->   (partita' = 0) & (contatore' = 0) &
                              (g' = cambio_giocatore);


[] faseDue  & guardiaDif & g = 1 & sommaAttDaGuno > 0->

     AttaccabileS1DaGuno/sommaAttDaGuno : (dif' = 1)+


     AttaccabileS2DaGuno/sommaAttDaGuno : (dif' = 2)+
```

```
     AttaccabileS3DaGuno/sommaAttDaGuno : (dif' = 3)+

     AttaccabileS4DaGuno/sommaAttDaGuno : (dif' = 4)+

     AttaccabileS5DaGuno/sommaAttDaGuno : (dif' = 5)+

     AttaccabileS6DaGuno/sommaAttDaGuno : (dif' = 6);
[]   faseDue  & guardiaDif & g = 1 &
     sommaAttDaGuno = 0 ->    (partita' = 0) & (contatore' = 0) &
                              (g' = cambio_giocatore);


//transizioni che si escludono a vicenda, solo quella
corrispondente al territorio che si dovrà difendere sarà
selezionata.
//da questa verrà fatta in autonomia (almeno per ora) la scelta
del territorio dal quale verrà sferrato l'attacco (essendocene
potenzialmente più di uno,
//esattamente da un min di 1 a un max di 3)
//la probabilità con la quale sarà selezionato il territorio dal
quale si sferrerà l'attacco è definita dalla frazione [casi
positivi/casi possibili]
//ovvero [potenziale territorio attaccante/somma dei territori dai
quali può essere attaccato]

[]   faseDue  & dif = 1 ->    cs2/(cs2 + cs3) : (att' = 2) &
                              (partita' = 3) +

                              cs3/(cs2 + cs3) : (att' = 3) &
                              (partita' = 3);

[]   faseDue  & dif = 2 ->    cs1/(cs1 + cs4 + cs5):(att' = 1) &
                              (partita' = 3) +

                              cs4/(cs1 + cs4 + cs5):(att' = 4) &
                              (partita' = 3) +

                              cs5/(cs1 + cs4 + cs5) : (att' = 5) &
                              (partita' = 3);

[]   faseDue  & dif = 3 ->    cs1/(cs1 + cs4 + cs5) : (att' = 1) &
                              (partita' = 3)  +

                              cs4/(cs1 + cs4 + cs5) : (att' = 4) &
                              (partita' = 3) +

                              cs5/(cs1 + cs4 + cs5) : (att' = 5) &
                              (partita' = 3);
```

```
[]   faseDue  & dif = 4 ->    cs2/(cs2 + cs3 + cs6) : (att' = 2) &
                             (partita' = 3) +

                             cs3/(cs2 + cs3 + cs6) : (att' = 3) &
                             (partita' = 3) +

                             cs6/(cs2 + cs3 + cs6) : (att' = 6) &
                             (partita' = 3);


[]   faseDue  & dif = 5 ->    cs2/(cs2 + cs3 + cs6) : (att' = 2) &
                             (partita' = 3) +

                             cs3/(cs2 + cs3 + cs6) : (att' = 3) &
                             (partita' = 3) +

                             cs6/(cs2 + cs3 + cs6) : (att' = 6) &
                             (partita' = 3);

[]   faseDue  & dif = 6 ->    cs4/(cs4 + cs5) : (att' = 4) &
                             (partita' = 3) +

                             cs5/(cs4 + cs5) : (att' = 5) &
                             (partita' = 3);

//FASE TRE: si svolge la BATTAGLIA
//con un dado solo le probabilità di vincere lo scontro sono così
ripartite: attacco = 41.7 | difesa = 58.3

//1 vs 2
[] faseTre & dif = 1 & att = 2 ->  0.417 : (c1' = c1 -1) &
                                  (partita' = 4 ) +

                                  0.583 : (c2' = c2 -1) &
                                  (partita' = 4 );

//1 vs 3
[]   faseTre &  dif = 1 & att = 3 ->    0.417 : (c1' = c1 -1) &
                                       (partita' = 4 ) +

                                       0.583 : (c3' = c3 -1) &
                                       (partita' = 4 );

//2 vs 1
[]   faseTre  & dif = 2 & att = 1 ->    0.417 : (c2' = c2 -1) &
                                       (partita' = 4 ) +

                                       0.583 : (c1' = c1 -1) &
                                       (partita' = 4 );
```

```
//2 vs 4
[]   faseTre  & dif = 2 & att = 4 ->    0.417 : (c2' = c2 -1)  &
                                        (partita' = 4 ) +

                                        0.583 : (c4' = c4 -1) &
                                        (partita' = 4 );

//2 vs 5
[]   faseTre  & dif = 2 & att = 5 ->    0.417 : (c2' = c2 -1) &
                                        (partita' = 4 ) +

                                        0.583 : (c5' = c5 -1) &
                                        (partita' = 4 );
//3 vs 1
[]   faseTre  & dif = 3 & att = 1 ->    0.417 : (c3' = c3 -1) &
                                        (partita' = 4 ) +

                                        0.583 : (c1' = c1 -1) &
                                        (partita' = 4 );

//3 vs 4
[]   faseTre  & dif = 3 & att = 4 ->    0.417 : (c3' = c3 -1) &
                                        (partita' = 4 ) +

                                        0.583 : (c4' = c4 -1) &
                                        (partita' = 4 );

//3 vs 5
[]   faseTre  & dif = 3 & att = 5 ->    0.417 : (c3' = c3 -1) &
                                        (partita' = 4 ) +

                                        0.583 : (c5' = c5 -1) &
                                        (partita' = 4 );

//4 vs 2
[]   faseTre  & dif = 4 & att = 2 ->    0.417 : (c4' = c4 -1) &
                                        (partita' = 4 ) +

                                        0.583 : (c2' = c2 -1) &
                                        (partita' = 4 );

//4 vs 3
[]   faseTre  & dif = 4 & att = 3 ->    0.417 : (c4' = c4 -1) &
                                        (partita' = 4 ) +

                                        0.583 : (c3' = c3 -1) &
                                        (partita' = 4 );

//4 vs 6
[]   faseTre  & dif = 4 & att = 6 ->    0.417 : (c4' = c4 -1) &
                                        (partita' = 4 ) +
```

```
                                        0.583 : (c6' = c6 -1) &
                                        (partita' = 4 );

//5 vs 2
[]   faseTre  & dif = 5 & att = 2 ->    0.417 : (c5' = c5 -1) &
                                        (partita' = 4 ) +

                                        0.583 : (c2' = c2 -1) &
                                        (partita' = 4 );

//5 vs 3
[]   faseTre  & dif = 5 & att = 3 ->    0.417 : (c5' = c5 -1) &
                                        (partita' = 4 ) +

                                        0.583 : (c3' = c3 -1) &
                                        (partita' = 4 );

//5 vs 6
[]   faseTre  & dif = 5 & att = 6 ->    0.417 : (c5' = c5 -1) &
                                        (partita' = 4 ) +

                                        0.583 : (c6' = c6 -1) &
                                        (partita' = 4 );

//6 vs 4
[]   faseTre  & dif = 6 & att = 4 ->    0.417 : (c6' = c6 -1) &
                                        (partita' = 4 ) +

                                        0.583 : (c4' = c4 -1) &
                                        (partita' = 4 );

//6 vs 5
[]   faseTre  & dif = 6 & att = 5 ->    0.417 : (c6' = c6 -1) &
                                        (partita' = 4 ) +

                                        0.583 : (c5' = c5 -1) &
                                        (partita' = 4 );

//NON CI SONO STATE CONQUISTE
[]   faseQuattro & dif != 0 & att != 0 &
!guardiaTerritorioConquistato  ->(dif' = 0) & (att' = 0);


//EVENTUALE SPOSTAMENTO

//difesa eliminata
[] faseQuattro & att = 2 & c1 = 0 ->(c1' = c1 + 1) & (c2'=c2-1) &
                                        (s1' = g) & (dif' = 0) &
                                        (att' = 0);
```

```
//1 vs 3
//difesa eliminata
[] faseQuattro & att = 3 & c1 = 0 ->(c1' = c1 + 1) & (c3'=c3-1) &
                                    (s1' = g) & (dif' = 0) &
                                    (att' = 0);


//2 vs 1
//difesa eliminata
[] faseQuattro  & att = 1 & c2 = 0 ->(c2' = c2 + 1) & (c1'=c1-1)
& (s2' = g) & (dif' = 0) & (att' = 0);




//2 vs 4
//difesa eliminata
[] faseQuattro & att = 4 & c2 = 0 ->(c2' = c2 + 1) & (c4'=c4-1) &
                                    (s2' = g) & (dif' = 0) &
                                    (att' = 0);


//2 vs 5
//difesa eliminata
[] faseQuattro  & att = 5 & c2 = 0 ->(c2' = c2 + 1) & (c5'=c5-1) &
                                    (s2' = g) & (dif' = 0) &
                                    (att' = 0);


//3 vs 1
//difesa eliminata
[] faseQuattro  & att = 1 & c3 = 0 ->(c3' = c3 + 1) & (c1'=c1-1) &
                                    (s3' = g) & (dif' = 0) &
                                    (att' = 0);



//3 vs 4
//difesa eliminata
[] faseQuattro  & att = 4 & c3 = 0 ->(c3' = c3 + 1) & (c4'=c4-1) &
                                    (s3' = g) & (dif' = 0) &
                                    (att' = 0);



//3 vs 5
//difesa eliminata
[] faseQuattro & att = 5 & c3 = 0 ->(c3' = c3 + 1) & (c5'=c5-1) &
                                    (s3' = g) & (dif' = 0) &
```

```
                                        (att' = 0);


//4 vs 2
//difesa eliminata
[] faseQuattro & att = 2 & c4 = 0 ->(c4' = c4  + 1) & (c2'=c2-1) &
                                        (s4' = g) & (dif' = 0) &
                                        (att' = 0);


//4 vs 3
//difesa eliminata
[] faseQuattro  & att = 3 & c4 = 0 ->(c4' = c4 + 1) & (c3'=c3-1) &
                                        (s4' = g) & (dif' = 0) &
                                        (att' = 0);


//4 vs 6
//difesa eliminata
[] faseQuattro  & att = 6 & c4 = 0 ->(c4' = c4 + 1) & (c6'=c6-1) &
                                        (s4' = g) & (dif' = 0) &
                                        (att' = 0);


//5 vs 2
//difesa eliminata
[] faseQuattro & att = 2 & c5 = 0 ->(c5' = c5 + 1) & (c2'=c2-1) &
                                        (s5' = g) & (dif' = 0) &
                                        (att' = 0);


//5 vs 3
//difesa eliminata
[] faseQuattro & att = 3 & c5 = 0 ->(c5' = c5 + 1) &  (c3'=c3-1) &
                                        (s5' = g) & (dif' = 0) &
                                        (att' = 0);


//5 vs 6
//difesa eliminata
[] faseQuattro  & att = 6 & c5 = 0 ->(c5' = c5 + 1) & (c6'=c6-1) &
                                        (s5' = g) & (dif' = 0) &
                                        (att' = 0);
```

```
//6 vs 4
//difesa eliminata
[] faseQuattro  & att = 4 & c6 = 0 ->(c6' = c6 + 1) & (c4'=c4-1) &
                                     (s6' = g) & (dif' = 0) &
                                     (att' = 0);



//6 vs 5
//difesa eliminata
[] faseQuattro & att = 5 & c6 = 0 ->(c6' = c6 + 1) &  (c5'=c5-1) &
                                     (s6' = g) & (dif' = 0) &
                                     (att' = 0);



//CONTROLLO SE LA PARTITA E' TERMINATA
[] faseQuattro  & dif = 0 & att = 0 & partitaFinita->(partita'=5);


//DECIDO SE ATTACCARE DI NUOVO O SE PASSARE IL TURNO

[]   faseQuattro  & dif = 0 &
                    att = 0 &
               !partitaFinita ->        0.5 : (partita' = 1) &
                                        (contatore' = 1) +

                                        0.5 : (partita' = 0) &
                                        (contatore' = 0) &
                                        (g' = cambio_giocatore);



endmodule
```

```
DUNGEONS CRAWL

dtmc

//la partita è in corso
formula giocoInCorso = ( (stanza > 0) & (stanza < maxStanze) &
                         (!vinconoUmani) & (!vinconoMostri) );

formula ultimaStanza = ( stanza = maxStanze );


//conta stanze per esperimenti
const int contaStanze;

//i giocatori sono tutti morti
formula vinconoMostri = ( (stanza < maxStanze) &
                         ( max(g1 , g2 , g3 , g4 ) = 0 ) ) |
                         ( vinceMostroFinale );

//i mostri sono stati tutti sconfitti e le stanze sono finite
formula vinconoUmani =  (vinconoFinaleG_i);


//tutti i giocatori arrivano salvi all'ultima stanza
formula tuttiSalvi = ( min(feriteG1, feriteG2,
                         feriteG3, feriteG4) > 0 );

//formula per inizializzazione delle stanze
formula NONinizializzata = (mostri = -1);

//mostro fiale NON inizializzato
formula mostroFinaleNONinizializzato = (mostroFinale = 0);

//mostro finale inizializzato
formula mostroFinaleInizializzato = (mostroFinale > 0);

//la stanza in cui si trovano i giocatori è stata ripulita dai
mostri
formula stanzaRipulita = (stanza <= (maxStanze - 1) ) &
                         (mostri = 0);

//etichette varie
label "StanzaRipulita" = stanzaRipulita;

label "vinconoImostri" = vinconoMostri;

label "vinconoGliUmani" = vinconoUmani;


//la stanza è stata riempita di mostri da picchiare
formula inizializzata = (mostri > 0) ; //& (trolls > 0);
```

```
//num max di ferite accumulabili da ogni singolo giocatore
const int maxFerite;

//probabilità con la quale si sceglie di attaccare o passare il
turno al giocatore successivo
const double p;

//per ora ogni giocatore può rifiutarsi di combattere un numero
max di volte stabilito prima di iniziare a giocare.
//scelgo di azzerare il conteggio dei "rifiuti" a combattere
soltanto una volta ripulita la stanza da tutti i mostri

//descrizione dei giocatori: ogni giocatore è composto da una
quintupla e da una formula. La quintupla è formata da
//tre variabili relative ai combattimenti e una relativa allo
stato del giocatore (attivo/NON attivo)
//La var g(i)combatte è un "bottone" che serve sia per gestire
l'azione dei singoli giocatori, sia per gestire l'eventuale
//danno (ferita) che un giocatore può subire. Prendendo valore 1
infatti andrà a determinare la probabilità che il giocatore
//che sta combattendo sia quello che subisce l'eventuale ferita,
dal momento che tutti gli altri avranno valore 0.
//La var g(i)NONcombatte serve per evitare un loop infinito nel
quale ogni giocatore si rifiuta di combattere e non succede
//mai niente. Questo si potrebbe pensare di farlo anche
infliggendo un danno a tutti i giocatori qualora si rifiutassero
//in blocco di combattere. Personalmente preferisco la soluzione
già adottata nel codice.
//punteggioFinaleG(i) sta a indicare il punteggio totalizzato dal
giocatore nel corso dello scontro con il mostro finale.
//La formula serve per determinare se un giocatore ha o meno
esaurito i punti ferita a sua disposizione.

formula sommaGi = (g1 + g2 + g3 + g4);

//numero di giocatori superstiti
const int superstiti;
//-------------------------------------------------------------
//G1
global g1 : [0..1] init 1;
global feriteG1 : [0..maxFerite] init maxFerite;
global g1combatte : [0..1] init 0;
global punteggioFinaleG1 : [0..12] init 0;

formula g1Vivo = (feriteG1 > 1);

//-------------------------------------------------------------

//G2
```

```
global g2 : [0..1] init 1;
global feriteG2 : [0..maxFerite] init maxFerite;
global g2combatte : [0..1] init 0;
global punteggioFinaleG2 : [0..12] init 0;

formula g2Vivo = (feriteG2 > 1);

//-----------------------------------------------------------------

//G3
global g3 : [0..1] init 1;
global feriteG3 : [0..maxFerite] init maxFerite;
global g3combatte : [0..1] init 0;
global punteggioFinaleG3 : [0..12] init 0;

formula g3Vivo = (feriteG3 > 1);

//-----------------------------------------------------------------

//G4
global g4 : [0..1] init 1;
global feriteG4 : [0..maxFerite] init maxFerite;
global g4combatte : [0..1] init 0;
global punteggioFinaleG4 : [0..12] init 0;

formula g4Vivo = (feriteG4 > 1);

//-----------------------------------------------------------------

//dadi utilizzati per i combattimenti

global dado1_2 : [0..6] init 0;



global dadoExtra : [0..6] init 0;

global dadoMostro : [0..6] init 0;



//var utilizzate per gestire la progressione degli scontri

global combattimento : [0..2] init 0;

global difesaMostro : [0..2] init 0;

//formule per determinare il vincitore del combattimento

formula offesaFinaleG_i = (punteggioFinaleG1 + punteggioFinaleG2 +
                           punteggioFinaleG3 +
```

```
                              punteggioFinaleG4) + (dadoExtra);

//danno inflitto dal mostro
formula difesa_Mostro = dadoMostro;


//vince il giocatore
formula vinceG_i = (dado1_2 >= difesa_Mostro);

//vince il mostro
formula vinceMostro = (dado1_2 < difesa_Mostro);

//i giocatori vincono contro il mostro finale
formula vinconoFinaleG_i = (ultimaStanza) & (mossa = 5) &
                              (offesaFinaleG_i >= mostroFinale);

//il mostro finale batte i giocatori super-istiti
formula vinceMostroFinale = ( ultimaStanza) & (mossa = 5) &
                              (offesaFinaleG_i < mostroFinale);

//num max di stanze "esplorabili"
const int maxStanze;

//-----------------------------------------------------------------

module M1

mostri : [-1..8] init -1;

mostroFinale : [-1..24] init 0;

mossa : [0..6] init 1;



sfidaFinale : [-1..1] init 0;

stanza : [-1..maxStanze] init 0;

//primo passo per far entrare i giocatori in una stanza infestata
dai mostri (eliminabile)
[] stanza = 0 -> (stanza' = stanza + 1);


[] giocoInCorso & NONinizializzata ->

                          0.25 : (mostri' = 4) +

                          0.25 : (mostri' = 5) +

                          0.25 : (mostri' = 6) +
```

```
                    0.25 : (mostri' = 7) ;

//-------------------------------------------------------------



//G1
//scelta
[] giocoInCorso & inizializzata & mossa = 1 & ( g1Vivo | feriteG1
= 1)  & combattimento = 0 & difesaMostro = 0 ->(g1combatte' = 1) &
                                               (combattimento' = 1) &
                                               (mossa' = 2);



//g1 ferite esaurite
[] giocoInCorso & inizializzata & mossa = 1 & feriteG1 = 0 &
combattimento = 0 & difesaMostro = 0 -> (mossa' = 2);

//-------------------------------------------------------------

//G2
//scelta
[] giocoInCorso & inizializzata & mossa = 2 & ( g2Vivo | feriteG2
= 1) & combattimento = 0 & difesaMostro = 0 -> (g2combatte' = 1) &
                                               (combattimento' = 1) &
                                               (mossa' = 3);

//g2 ferite esaurite
[] giocoInCorso & inizializzata & mossa = 2 & feriteG2 = 0 &
combattimento = 0 & difesaMostro = 0 -> (mossa' = 3);

//-------------------------------------------------------------

//G3
//scelta
[] giocoInCorso & inizializzata & mossa = 3 & ( g3Vivo | feriteG3
= 1) & combattimento = 0 & difesaMostro = 0 -> (g3combatte' = 1) &
                                               (combattimento' = 1) &
                                               (mossa' = 4);

//g3 ferite esaurite
[] giocoInCorso & inizializzata & mossa = 3 & feriteG3 = 0 &
combattimento = 0 & difesaMostro = 0 -> (mossa' = 4);

//-------------------------------------------------------------

//G4
//scelta
```

```
[] giocoInCorso & inizializzata & mossa = 4 & ( g4Vivo | feriteG4
= 1) & combattimento = 0 & difesaMostro = 0 -> (g4combatte' = 1) &
                                                (combattimento' = 1) &
                                                (mossa' = 1);

//g4 ferite esaurite
[] giocoInCorso & inizializzata & mossa = 4 & feriteG4 = 0 &
combattimento = 0 & difesaMostro = 0 -> (mossa' = 1);

//-----------------------------------------------------------------

//combattimento
[] giocoInCorso & inizializzata & combattimento = 1 & difesaMostro
= 0 ->    1/36 : (dado1_2' = 1) & (combattimento' = 2) +

     3/36 : (dado1_2' = 2) & (combattimento' = 2) +

     5/36 : (dado1_2' = 3) & (combattimento' = 2) +

     7/36 : (dado1_2' = 4) & (combattimento' = 2) +

     9/36 : (dado1_2' = 5) & (combattimento' = 2) +

     11/36 : (dado1_2' = 6) & (combattimento' = 2) ;


[] giocoInCorso & inizializzata & combattimento = 2 & difesaMostro
= 0 ->    1/6 : (dadoMostro' = 1) & (difesaMostro' = 1) +

          1/6 : (dadoMostro' = 2) & (difesaMostro' = 1) +

          1/6 : (dadoMostro' = 3) & (difesaMostro' = 1) +

          1/6 : (dadoMostro' = 4) & (difesaMostro' = 1) +

          1/6 : (dadoMostro' = 5) & (difesaMostro' = 1) +

          1/6 : (dadoMostro' = 6) & (difesaMostro' = 1) ;


//-----------------------------------------------------------------


//determinare chi ha vinto il combattimento


//VINCE UN GIOCATORE
[] giocoInCorso & inizializzata & combattimento = 2 & difesaMostro
= 1 & vinceG_i & mostri >= 1 ->

     (mostri' = mostri - 1) &
```

```
      (combattimento' = 0) &

      (difesaMostro' = 0) &

      (dado1_2' = 0) &

      (dadoMostro' = 0) &


      (g1combatte' = 0) &

      (g2combatte' = 0) &

      (g3combatte' = 0) &

      (g4combatte' = 0) ;

//VINCE IL MOSTRO
[] giocoInCorso & inizializzata & combattimento = 2 & difesaMostro
= 1 & vinceMostro & g1combatte = 1 & g1Vivo ->

       (feriteG1' = feriteG1 - 1) &

       (combattimento' = 0) &

       (difesaMostro' = 0) &

       (dado1_2' = 0) &

       (dadoMostro' = 0) &

       (g1combatte' = 0);

//il giocatore G1 ha esaurito i suoi punti ferita
[] giocoInCorso & inizializzata & combattimento = 2 & difesaMostro
= 1 & vinceMostro & g1combatte = 1 & feriteG1 = 1 ->

           (feriteG1' = feriteG1 - 1) & (g1' = 0) &

           (combattimento' = 0) &

           (difesaMostro' = 0) &

           (dado1_2' = 0) &

           (dadoMostro' = 0) &

           (g1combatte' = 0);
```

```
[] giocoInCorso & inizializzata & combattimento = 2 & difesaMostro
= 1 & vinceMostro & g2combatte = 1 & g2Vivo ->

      (feriteG2' = feriteG2 - 1) &

      (combattimento' = 0)&

      (difesaMostro' = 0) &

      (dado1_2' = 0) &

      (dadoMostro' = 0) &

      (g2combatte' = 0);

//il giocatore G2 ha esaurito i suoi punti ferita
[] giocoInCorso & inizializzata & combattimento = 2 & difesaMostro
= 1 & vinceMostro & g2combatte = 1 & feriteG2 = 1 -> (feriteG2' =
feriteG2 - 1) & (g2' = 0) &

          (combattimento' = 0) &

          (difesaMostro' = 0) &

          (dado1_2' = 0) &

          (dadoMostro' = 0) &

          (g2combatte' = 0);


[] giocoInCorso & inizializzata & combattimento = 2 & difesaMostro
= 1 & vinceMostro & g3combatte = 1 & g3Vivo ->

      (feriteG3' = feriteG3 - 1) &

      (combattimento' = 0)&

      (difesaMostro' = 0) &

      (dado1_2' = 0) &

      (dadoMostro' = 0) &

      (g3combatte' = 0);

//il giocatore G3 ha esaurito i suoi punti ferita
[] giocoInCorso & inizializzata & combattimento = 2 & difesaMostro
= 1 & vinceMostro & g3combatte = 1 & feriteG3 = 1 ->

          (feriteG3' = feriteG3 - 1) & (g3' = 0) &
```

```
          (combattimento' = 0) &

          (difesaMostro' = 0) &

          (dado1_2' = 0) &

          (dadoMostro' = 0) &

          (g3combatte' = 0);


[] giocoInCorso & inizializzata & combattimento = 2 & difesaMostro
= 1 & vinceMostro & g4combatte = 1 & g4Vivo ->

      (feriteG4' = feriteG4 - 1) &

      (combattimento' = 0)&

      (difesaMostro' = 0) &

      (dado1_2' = 0) &

      (dadoMostro' = 0) &

      (g4combatte' = 0);

//il giocatore G4 ha esaurito i suoi punti ferita
[] giocoInCorso & inizializzata & combattimento = 2 & difesaMostro
= 1 & vinceMostro & g4combatte = 1 & feriteG4 = 1 ->

          (feriteG4' = feriteG4 - 1) & (g4' = 0) &

          (combattimento' = 0) &

          (difesaMostro' = 0) &

          (dado1_2' = 0) &

          (dadoMostro' = 0) &

          (g4combatte' = 0);
//-----------------------------------------------------------------

//si passa alla stanza successiva

[] giocoInCorso & stanzaRipulita & combattimento = 0 &
difesaMostro = 0 -> (stanza' = stanza + 1) &
                    (mostri' = -1) &
                    (mossa' = 1);
```

```
//-------------------------------------------------------------
//INIZIA LA SFIDA CON IL MOSTRO FINALE

[] ultimaStanza & mostroFinaleNONinizializzato →

                                1/1296 : (mostroFinale' = 4) +
                                4/1296 : (mostroFinale' = 5) +
                                10/1296 : (mostroFinale' = 6) +
                                20/1296 : (mostroFinale' = 7) +
                                35/1296 : (mostroFinale' = 8) +
                                56/1296 : (mostroFinale' = 9) +
                                80/1296 : (mostroFinale' = 10) +
                                104/1296 : (mostroFinale' = 11) +
                                125/1296 : (mostroFinale' = 12) +
                                140/1296 : (mostroFinale' = 13) +
                                146/1296 : (mostroFinale' = 14) +
                                140/1296 : (mostroFinale' = 15) +
                                125/1296 : (mostroFinale' = 16) +
                                104/1296 : (mostroFinale' = 17) +
                                80/1296 : (mostroFinale' = 18) +
                                56/1296 : (mostroFinale' = 19) +
                                35/1296 : (mostroFinale' = 20) +
                                20/1296 : (mostroFinale' = 21) +
                                10/1296 : (mostroFinale' = 22) +
                                4/1296 : (mostroFinale' = 23) +
                                1/1296 : (mostroFinale' = 24);



//G1
[] ultimaStanza & mostroFinaleInizializzato & g1 = 1 & mossa = 1->

     1/36 : (punteggioFinaleG1' = 1) & (mossa' = 2) +

     3/36 : (punteggioFinaleG1' = 2) & (mossa' = 2) +

     5/36 : (punteggioFinaleG1' = 3) & (mossa' = 2) +

     7/36 : (punteggioFinaleG1' = 4) & (mossa' = 2) +

     9/36 : (punteggioFinaleG1' = 5) & (mossa' = 2) +

     11/36 : (punteggioFinaleG1' = 6) & (mossa' = 2) ;

[] ultimaStanza & mostroFinaleInizializzato & g1 = 0 & mossa = 1->

          (mossa' = 2);
```

```
//G2
[] ultimaStanza & mostroFinaleInizializzato & g2 = 1 & mossa = 2->

     1/36 : (punteggioFinaleG2' = 1) & (mossa' = 3) +

     3/36 : (punteggioFinaleG2' = 2) & (mossa' = 3) +

     5/36 : (punteggioFinaleG2' = 3) & (mossa' = 3) +

     7/36 : (punteggioFinaleG2' = 4) & (mossa' = 3) +

     9/36 : (punteggioFinaleG2' = 5) & (mossa' = 3) +

     11/36 : (punteggioFinaleG2' = 6) & (mossa' = 3) ;


[] ultimaStanza & mostroFinaleInizializzato & g2 = 0 & mossa = 2->

     (mossa' = 3);


//G3
[] ultimaStanza & mostroFinaleInizializzato & g3 = 1 & mossa = 3->
     1/36 : (punteggioFinaleG3' = 1) & (mossa' = 4) +

     3/36 : (punteggioFinaleG3' = 2) & (mossa' = 4) +

     5/36 : (punteggioFinaleG3' = 3) & (mossa' = 4) +

     7/36 : (punteggioFinaleG3' = 4) & (mossa' = 4) +

     9/36 : (punteggioFinaleG3' = 5) & (mossa' = 4) +

     11/36 : (punteggioFinaleG3' = 6) & (mossa' = 4) ;


[] ultimaStanza & mostroFinaleInizializzato & g3 = 0 & mossa = 3->

     (mossa' = 4);


//G4
[] ultimaStanza & mostroFinaleInizializzato & g4 = 1 & mossa = 4->

     1/36 : (punteggioFinaleG4' = 1) & (mossa' = 5) +

     3/36 : (punteggioFinaleG4' = 2) & (mossa' = 5) +

     5/36 : (punteggioFinaleG4' = 3) & (mossa' = 5) +

     7/36 : (punteggioFinaleG4' = 4) & (mossa' = 5) +
```

```
     9/36 : (punteggioFinaleG4' = 5) & (mossa' = 5) +

     11/36 : (punteggioFinaleG4' = 6) & (mossa' = 5) ;

[] ultimaStanza & mostroFinaleInizializzato & g4 = 0 & mossa = 4->

     (mossa' = 5);




//i giocatori sconfiggono il mostro finale
[] vinconoFinaleG_i ->   (mossa' = 0) & (stanza' = -1);


//il mostro finale sconfigge i giocatori
[] vinceMostroFinale ->  (mossa' = 0) & (stanza' = -1);


endmodule
```