GRP_11: Modelling Street Fighter

Marcus Chiang (19MHZC)

Jalal Ghaus (20SJG)

Sophie Ellwood (19SE)

Pratyush Kaintura (20PK1)

Course Modelling Project

CISC/CMPE 204

Logic for Computing Science

October 29, 2021

Abstract

Our project is designed to loosely model the game Street fighter. A player will be given a situation where they must make the best possible move based on the propositions that are given. We decided to stick to one scenario at a time to imitate a puzzle and avoid the complexity of back and forth attacks.

The game will end when the principle player is hit or hits their opponent (who behaves as an NPC and serves to set up the attacks).

Propositions

Many of our propositions span between each player, the specified player is either 1 or 2 and is followed by an underscore. Player 1 punching would be P_1. I'm going to list the propositions without the player notation because they are the same between each.

- D denotes a player being damaged

- P denotes a player performing a punch (lands if one space between)

- K denotes a player performing a kick (lands if two spaces between)

- T denotes a player performing a throw(lands if adjacent)

- H denotes a player performing a hadouken(fireball projectile, lands if three spaces between)

- SHORYU denotes a player performing a shoryuken (invincible attack, lands if adjacent)

All of the above propositions have a defined range that is also defined with a distance system of propositions.

- W means a player has won a particular snapshot

- D means a player has been damaged

- B means a player is blocking

- JUMP means a player has performed a jump

- NEUTRAL means a player is in a neutral state or has returned to a neutral state after a particular interaction

- WHIFF means a player has whiffed an attack, it did not connect because the distance between players exceeded the attacks range

The following propositions relate to our positioning system and how the previous attacks either miss or hit.

- We have a loop that creates propositions for players being in positions 0 through 5, for a total of 6 positions. They range from Pposition0 to Pposition5.

- fourSpacesBetween means that using the position and or conditions ($P1position0 \wedge P2position5$) or the converse, we have deduced that there are four spaces between the two players

- threeSpacesBetween means that there are three spaces between the players

- twoSpacesBetween means that there are two spaces between the players

- oneSpacesBetween means that there is one space between the players

- adjacent means that there is no space between the players and they are right next to each other

We also have propositions for inputs. For example, $(2 \wedge 3 \wedge 6 \wedge P) \rightarrow$ H; this means that down, down-right, right, and punch will imply a hadouken. However, we have omitted these for the draft since we are unsure if we want to keep this feature.

Constraints

We realize that we have a lot of constraints and need to continue to work on them to
a) make sure they cover all situations and
b) are optimized in the code.
We have constraints that moves cannot be used together. This is a lot to simply list out so we'd like to find a more efficient way of describing them.

- Both players cannot win $\neg(W_1 \neg W_2)$

- Player 1 can't block and get hit at the same time $\neg(D_1/wedgeB_1)$

- Punch and kick from a player result in a throw $((I_P \wedge I_K) \implies T_1)$

- player 1 nor 2 may perform a kick and punch at the same time $((P_1 \wedge K_1)$

- players may jump to not get hit form attacks $(H_1 \wedge JUMP_2)$

- Throw Break $(((T_1 \wedge T_2) \wedge (adjacent)) \implies bothNeutral)$

- Throwing a blocking opponent works $(((B_1) \wedge (T_2)) \wedge adjacent$

- If player punches but are not adjacent to each other nothing happens $((P_1 \wedge \neg adjacent) \implies (WHIFF_1))$

- nothing happens if player blocks an attack $(((P_1 \vee K_1 \vee H_1 \vee SHORYU) \wedge (B \vee_2)) \implies bothNeutral)$

- whiff occurs if players attack are within 2,3,4 spaces $((twoSpacesBetween \vee threeSpacesBetween \vee fourSpacesBetween \wedge K_1) \implies (WHIFF_1))$

- If player 1 uses H and there is 4 spaces between player 2 then player 1 whiffs $(((H_1) \wedge (fourSpacesBetween)) \implies (WHIFF_1)$

- If player 1 throws and player 2 is not adjacent then player 1 whiffs $((T_1) \wedge (adjacent)) \implies (WHIFF_1)$

- If player 1 is blocking and player 2 does an attack then nothing happens $((P_2 \vee K_2 \vee H_2 \vee SHORYU) \wedge (B_1) \implies (bothNeutral))$

Model Exploration

Our model will be satisfied by obtaining an outcome that "wins". For example, if an attack is launched against the player, the model will be satisfied by producing all the actions that can successfully defend against this attack.

This requires us to modify the truth value of propositions that represent elements in the gamestate. Different combinations of elements represent different scenarios and snapshots of an actual game of street fighter.

For example, negating a constraint would be like taking rule out of the game (ex.allowing a player to move forward and backward at the same time) and negating an action would be the absence of that action (ex. not blocking an attack).

First-Order Extension

Our model is satisfied by attacking or defending correctly. It is possible that there can be many attacks and only a few defences that work in any given situation. In terms of universal quantifiers:

- We could say that for all possible attacks that use legs (all types of kicks, spins etc.) There exists a defence that blocks the attack.
- We could state that for all jumps, you can jump AND kick at the same time.

So we could update our propositions by grouping attacks into categories and using the universal and existential quantifiers to address specific moves within the group.
Using predicate logic would also simplify our constraints. We could state that for all x where some illegal action is performed, that this is not allowed.

If we wanted to further extend beyond the scope of our game, we could re-implement the health bar that omitted for the sake of simplicity. We could have $\forall p(M(p) \rightarrow M(d))$ meaning that for all punches, a medium punch would imply medium damage to the other player. We could have an integer variable representing the opponents life bar which would decrease if the proposition d evaluates to true. Therefore the amount of damage would be based on the predicate applied to the damage proposition.

Requested Feedback

We are thinking of adding user input for a keypad so based on what key is pressed, a certain action would be taken.
Should we have buttons for user inputs or does it complicate things too much? Is this project reasonably complex? Should we simplify or make it more complicated.
How can we optimize our constraints? (see constraint section above). We want to randomly generate scenarios.

We want to randomly generate propositions to set up each scenario Extend program to be able to play multiple times Omitting inputs for simplicity? right now we only have one snapshot, should we make it longer?

Jape Proofs

1: P1POSITION0, P2POSITION3	premises
2: (P1POSITION0∧P2POSITION3)→H2, R, (R∧H2)→Q	premises
3: P1POSITION0∧P2POSITION3	∧ intro 1.1,1.2
4: H2	→ elim 2.1,3
5: R∧H2	∧ intro 2.2,4
6: Q	→ elim 2.3,5

First Proof
Players are in positions right next to eachother, so we conclude adjacent.
Player 2 (NPC) is performing a punch (P2) and this means that we cannot block (negation of B) since they cannot perform two actions at once.
Player 1 needs to make the best possible move.
If Player 2 is not blocking, and player 1 performs a shoryuken, and they are adjacent, this means that the shoryuken connects concluding Player 2 is damaged. Player 2 being damaged implies Player 1 has successfully dealt with the situation and we have proven the best possible move.

1: P1POSITION2, P2POSITION3	premises
2: (P1POSITION2∧P2POSITION3)→ADJACENT	premise
3: T1, T2, ((T1∧T2)∧ADJACENT)→Q	premises
4: T1∧T2	∧ intro 3.1,3.2
5: P1POSITION2∧P2POSITION3	∧ intro 1.1,1.2
6: ADJACENT	→ elim 2,5
7: T1∧T2∧ADJACENT	∧ intro 4,6
8: Q	→ elim 3.3,7

Second Proof Player 1 is in position 1 and player 2 is in position 4, implying three spaces between them. Three spaces between is a constraint that dictates the range of hadouken. If player 1 jumps (as a defence) while a hadouken is being thrown, this is a solution to the situation. Player 1 is jumping, therefore we can deduce Q (successful move).

1: P1POSITION2, P2POSITION3	premises
2: (P1POSITION2∧P2POSITION3)→ADJACENT	premise
3: P1POSITION2∧P2POSITION3	∧ intro 1.1,1.2
4: ADJACENT	→ elim 2,3

Third Proof Player 1 is in position 0 and player 2 is in position 1, we can deduce that they are adjacent. Player 2 executes a throw, only works if adjacent (as shown in throw break proposition). If player 2 is trying to throw player 1, the solution is to break the throw by returning a throw of your own. A break implies that player 1 has successfully dealt with the situation, so we deduce Q (successful move).

$\wedge \quad \vee \quad \neg \quad \rightarrow \quad \forall \quad \exists$