Project Summary

Short summary of the project setting.

Our project aims to predetermine whether or not the player will lose a life if they build a line. The model will figure this out based on the current cursor position and cursor orientation (vertical or horizontal), as well as with a list of the current ball positions and velocities. It will also need a map of which cells have been captured already.

Propositions

List of the propositions used in the model, and their (English) interpretation.

Some aspects of the game that can be modeled as propositions:

Horizontal – The horizontal orientation is selected

Vertical – The vertical orientation is selected

Orientation (o) – Nests the two orientations into one single proposition. Could be either Horizontal or Vertical.

$\text{Mouse}_{(x, y, o)}$ – The cursor/mouse is located at (x,y) with an orientation O.

$\text{Captured}_{(x, y)}$ – A cell within the true game domain located at (x, y) is captured (i.e. becomes a black cell)

$\text{Position}_{(i, x, y, t)}$ – Ball i's position is located at cell (x, y) at time t

Direction (D) – Specifies all four cardinal directions, North, East, South and West.

$\text{Build}_{(x, y, D, t)}$ – A builder is building at the location (x, y)cin the direction specified by D, at time t.

Lost – The player loses a life while building a line

To handle the velocities of a ball, we will need to use the following predicates, that also follow the physics of the real game:

$\text{VelocityX}_{(i)}$ – An expression that indicates whether a ball i is moving is the positive or negative x direction.

$\text{VelocityY}_{(i)}$ – An expression that indicates whether a ball i is moving is the positive or negative y direction.

Constraints

List of constraint types used in the model and their (English) interpretation. You only need to provide one example for each constraint type: e.g., if you have constraints saying "cars have one colour assigned" in a car configuration setting, then you only need to show the constraints for a single car. Essentially, we want to see the pattern for all of the types of constraints, and not every constraint enumerated.

The cursor's orientation can only be either vertical or horizontal, but not both

$\forall o. \neg ((\text{Horizontal} \vee \text{Vertical}) \rightarrow (\text{Horizontal} \wedge \text{Vertical}))$

The position of a ball cannot coincide with the position of a captured cell

$\forall x.\ \forall y.\ \neg(\text{Position}_{(i,\,x,\,y,\,t)} \rightarrow \text{Captured}_{(x,\,y)})$

Only one horizontal or vertical pair of builders can be actively building at one given moment

$\forall \text{Build}.((\text{Build}_{(x,\,y,\,N,\,t)} \wedge \text{Build}_{(x,\,y,\,S,\,t)}) \vee (\text{Build}_{(x,\,y,\,E,\,t)} \wedge \text{Build}_{(x,\,y,\,W,\,t)})) \rightarrow \forall x.\ \forall y.\ (\neg\exists B.\ (\text{Build}_{(x,\,y,\,D\ t)}))$

When a ball collides with a builder, the player loses a life

$\forall \text{Position}.(\exists x.\exists y.(\text{Position}_{(i,\,x,\,y,\,t} \wedge \text{Build}_{(x,\,y,\,N,\,t)}) \rightarrow \text{Lost})$

Model Exploration

List all the ways that you have explored your model – not only the final version, but intermediate versions as well. See (C3) in the project description for ideas.

Our current model will output True or False depending on whether the player will lose a life or not after building a line. We had initially debated between this or whether the output was going to be the current number of lives the player has left. But we chose to go with the former because first, providing a binary output allows the player to understand their result with clarity. Secondly, a binary output is better suited to be modelled using propositional and predicate logic. It's important to note that we still keep track of the lives the user has left for game-functionality, however this information is not included in the output.

The next challenge we faced was figuring out the mechanics of how the program was going to determine if a ball was going to collide with a line being built. In JezzBall, the balls only move in 4 diagonals, meaning if the ball is travelling in the North-East direction, the x and y direction would both be positive. This allowed us to create these 2 propositions:

VelocityX$_{(i)}$ – An expression that's True if a ball 'i' is moving is the positive x direction and False if it's moving in the negative x direction.

VelocityY$_{(i)}$ –An expression that's True if a ball 'i' is moving is the positive y direction and False if it's moving in the negative y direction.

Jape Proof Ideas

List the ideas you have to build sequents & proofs that relate to your project.

Requested Feedback

Provide 2-3 questions you'd like the TA's and other students to comment on.

First-Order Extension

Describe how you might extend your model to a predicate logic setting, including how both the propositions and constraints would be updated. ***There is no need to implement this extension!***

We can use first-order extension to introduce additional predicates and quantifiers to handle more complex relationships and interactions. Here are some examples:

Proposition Extensions:

- **Time (T)**: Introduce a predicate to represent time, allowing you to model changes and interactions over time. For example, you can add "Time(t)" to indicate the current time step.
- **Collision (C)**: Create a predicate to handle collisions between balls and builders. For instance, "Collision(i, j, t)" can represent that ball i collides with builder j at time t.
- **Win (W)**: Add a predicate to denote when the player wins the game. "Win" can be true when certain conditions are met, such as capturing a specific number of cells.
- **Ball Deactivation (BAd)**: "BallDeactivation(i, t)" indicates that ball i is no longer in play at time t.
- **Builder Deactivation (BDe)**: "BuilderDeactivation(b, t)" denotes that builder b is no longer active at time t.
- **Builder Direction (BD)**: "BuilderDirection(b, t, D)" specifies that builder b is constructing a line in direction D at time t.
- **Cell Adjacency (CA**): "CellAdjacency(x1, y1, x2, y2)" expresses that cells (x1, y1) and (x2, y2) are adjacent.

Constraint Extensions:

- **Time Ordering**: To ensure that time steps are ordered correctly, you can use a universal quantifier. For example, "$\forall$ t. Time(t) $\rightarrow$ Time(t+1)" enforces that time progresses sequentially for all time steps.
- **Mutual Exclusivity**: To extend the constraint for cursor orientation while adding quantifiers, you can use a combination of universal and existential quantifiers. For example, "$\forall$ t. $\exists$ o. (Horizontal(t) $\vee$ Vertical(t)) $\rightarrow$ (Horizontal(t) $\wedge$ Vertical(t))" ensures that for all time steps, either the horizontal or vertical orientation is selected, but not both.
- **Ball Movements**: To enforce constraints related to ball movements, you can use a universal quantifier. For instance, "$\forall$ i. $\forall$ t. VelocityX(i, t) $\rightarrow$ (Position(i, x, y, t) $\wedge$ (Position(i, x+1, y, t) $\vee$ Position(i, x-1, y, t)))" ensures that for all balls and time steps, the ball's X velocity corresponds to its X position at the next time step.
- **Winning Condition**: For creating constraints related to winning conditions, you can use an existential quantifier. For example, "$\exists$ x1. $\exists$ y1. $\exists$ x2. $\exists$ y2. $\forall$ t. Win $\rightarrow$ (Captured(x1, y1, t) $\wedge$ Captured(x2, y2, t))" specifies that there exist positions (x1, y1) and (x2, y2) such that for all time steps, winning conditions are met when specific cells are captured.
- **Ball-Ball Collisions**: To extend collision constraints to handle interactions between balls while adding quantifiers, you can use a combination of universal and existential quantifiers. For example, "$\forall$ i. $\forall$ j. $\exists$ t. Collision(i, j, t) $\rightarrow$ $\neg$($\exists$ x. $\exists$ y. Position(i, x, y, t) $\wedge$ Position(j, x, y, t))" specifies that for all pairs of balls i and j, there exists a time t when a collision occurs, and at that time, they cannot occupy the same position.

Useful Notation

Feel free to copy/paste the symbols here and remove this section before submitting.

∧ ∨ ¬ → ∀ ∃