15-424/15-624 Recitation 9
Game Proofs with Street Fighter

1 Street Fighter

Image Sources:`https://www.pinterest.com/pin/348747564871355869/`, `http://streetfighter.com/characters/ryu/`

Figure 1: Left: Ryu (Angel), Right: Akuma (lit. Demon)

Today we're not only going to do proofs about games, we're also going to look at a new game! Street Fighter is a popular hybrid game franchise. It combines discrete decision making such as "Do I move my dude? Do I punch the other dude?" with continuous dynamics like "My dude is moving either because I moved it or because it got punched".

Today we'll consider a simplified model of Street Fighter. The playing field will be very similar to that for ping-pong: we consider a one-dimensional interval $[l, r]$, where the two fighters x_1 (Ryu, or Angel) and x_2 (Akuma, or Demon) both start between l and r. To avoid modeling hit points, we say a player loses if their dude gets knocked off the field: Angel loses if $x_1 < l$ and Demon loses if $x_2 > r$. Not only that, but recall that *every* scenario needs to have a winner, no draws allowed. So in the case of a draw (neither player knocks the other off the field), we say Angel loses. I suppose it's also possible that they knock each other off the field at the exact same time, which is also a draw, so Angel should lose then too.

Each player has two possible actions: punch the opponent or accelerate/brake themselves. Punching only works if you're close enough to hit the person, let's call that d_{hit}. The typical punch takes a very short period of time, so a good approximation of a punch is to discretely set the opponent's velocity to your *punch velocity* P_1 or P_2, whereas a good model of acceleration/braking is to discretely set the *acceleration* instead with your acceleration A_1 or A_2. Note we use the same acceleration whether we're braking in response to getting punched by our opponent or whether we're accelerating toward them. Not only is this

simpler, but intuitively braking and accelerating actually achieve the same objective in this game: getting further to the right (for Angel) or further to the left (for Demon). So now we've got a game where both acceleration and velocity can change discretely. Hopefully that will lead to interesting dynamics and interesting strategy! We're going to do a time-triggered model, where we run the loop again every time $t = T$.

We structure the game very similarly to our game from last week, however to make our lives easier, some of the physics is going to happen before the control, specifically collision detection happens before everything else. That leaves us with: (a) detect collisions (b) the Angel player, Ryu, makes his move with α_A, (c) the Demon Player, Akuma, makes his move with α_R and then (d) the physics α^P evolve continuously, and we think hard about whether they ought to be Angel or Demon.

$$\alpha = (\alpha_C; \alpha_R; \alpha_A; \alpha_P)^*$$

```
aC ==
  /* Collision detection: If you hit each other stop everything. */
  {?(x1 = x2 & v1 >= v2); v1 := 0; v2 := 0; a1 := 0; a2 := 0;}
++{?(true);}

aR ==
   /* You can punch if you're close enough. */
   {?(x2 - x1 <= d_hit); v2 := P1;}
   /* Instead you can always move toward the opponent */
++ {?(x2 > x1); a1 := A1;}
   /* Or do nothing if you want I guess */
++ {?(true);}


aA ==
   /* You can punch if you're close enough. */
   {?(x2 - x1 <= d_hit); v1 := -P2;}
   /* Instead you can always move toward the opponent */
++ {?(x2 > x1); a2 := -A2;}
   /* Or do nothing if you want I guess */
++ {?(true);}


aP ==
  /* Evolve continuously, or stop early if you collide */
  {x1' = v1, v1' = a1, x2' = v2, v2' = a2 & x1 <= x2}
```

2 Staying Alive: Strategy for Demon

For any interesting game, different players will win under different starting conditions. For any interesting game, it is also really wise if we start out thinking about a couple of *simple* conditions where we know a player should win before we attempt a general strategy. One simple case is when the fighters are equally powerful ($P_1 = P_2$) with equal running ability ($A_1 = A_2$) and the both start out in the center ($x_1 = x_2 = (l + r)/2$). In this case they're a perfectly even match, so it should end in a "draw" which, remember, means that Demon wins. For Demon, winning should mean proving that he always stays on the playing field - it doesn't actually matter whether he knocks Angel off or not.

$$\text{Pre}_W \equiv P_1 = P_2 \wedge A_1 = A_2 \wedge x_1 = x_2 \wedge x_1 = (l + r)/2$$

$$Pre \rightarrow [\alpha] x_2 \leq r$$

Strategy What's a good strategy for Demon in this scenario? In general, any time that a game looks really symmetric (think Nim), the Demon player can use a *mirroring* strategy: look at whatever the Angel player does and do the opposite. In some sense this is a *dumb* strategy: you will never try to actually **beat** Angel, you'll just make sure you play him into a stalemate and then win by default (see last week's discussion of filibustering loops). At the same time, it leads to nice simple proof arguments for Demon because mirroring strategies lend themselves well to invariant reasoning. Plus, if all you care about is whether or not you win, it won't help you to kick Angel's butt extra hard, you might as well just do what it takes to win.

In this case Angel has two moves: punch or walk. Will mirroring get us into our desired stalemate?

- Punching case: Here we follow the advice my mother gave me when I was small: If someone hits you, hit back. This will cause us to start moving away from the center equally fast. It's possible that we'll get knocked off the edge of the arena by the punch, but because we're mirroring Angel perfectly, he would get knocked off at the same time. This is a draw so we win. And if we don't get knocked off, then we're still stalemating.
- Moving case: My mother also told me not to *start* fights. If we accelerate when they accelerate, we will continue to match their trajectory. Also we might run into each other, but that's not considered a safety violation. Ryu and Akuma are tough and violence is honestly kind of the point of this recitation.
- Do-nothing case: In this case, if we were being *clever*, then we would maybe punch or accelerate and gain an edge over the opponent. However, the more complicated our strategy is, the more complicated our proof will be. We shouldn't needlessly limit our

imagination: going for the fancy strategy is great, but we should take baby steps and start with an easy strategy first. My mother doesn't know how to do proofs so she never gave me advice on this case, but luckily I figured it out myself.

Invariants As we can see, our whole strategy is based on mirroring: it's an invariant that the Angel and Demon players will mirror each other at all times. As we did in hybrid systems, we prove box properties by invariant reasoning: the Demon player has no idea how long Angel will run the loop, so they had better have an argument that works for all numbers of iterations, an invariant argument. Mathematically, how do we say that we're mirroring each other? That means our average position $(x_1 + x_2)/2$ is always the average of the endpoints l and r: $(l + r)/2$. Or, surprisingly, $x_1 + x_2 = l + r$! Aren't invariants so pretty sometimes!?

The invariant rule:

$$\text{ind}\ \frac{\Gamma \vdash J, \Delta \qquad J \vdash [\alpha]J \qquad J \vdash P}{\Gamma \vdash [\alpha]P, \Delta}$$

We're going to use some abbreviations to increase the chances of reading the proof. We define an invariant J:

$$J \equiv x_1 + x_2 = l + r \land v_1 + v_2 = 0 \land a_1 + a_2 = 0$$

and also use $\Gamma_{\texttt{const}}$ with its standard meaning of "all the constant formulas in the initial context Γ"

The proof starts out by applying the loop rule for games. The precondition and postcondition cases go through easily. As usual all the work is in the inductive step:

$$\text{ind}\ \frac{\mathbb{R}\ \dfrac{*}{x_1 + x_2 = l + r \vdash (x_2 > r \to x_1 < l)} \qquad \mathbb{R}\ \dfrac{*}{\Gamma \vdash x_1 + x_2 = l + r} \qquad \Gamma_{\texttt{const}}, J \vdash [\alpha_C; \alpha_R; \alpha_A^d; \alpha_P]J}{\Gamma \vdash [(\alpha_C; \alpha_R; \alpha_A^d; \alpha_P)](x_2 > r \to x_1 < l)}$$

For now we're going to ignore the collision avoidance case until later, first because the proof is already complicated enough, second because it will be a while until we need it. Just remember that collision avoidance gives us the nice assumption $x_1 = x_2 \to (v_1 = v_2 = 0 \land a_1 = a_2 = 0)$ and this will be helpful later.

Rigorous Ignoring It's not such a bad thing to ignore the collision detection right now because it turns out that it preserves the invariant J, so we could split up the proof like this:

$$\text{HM}\,\frac{J \vdash [\alpha_R;\alpha_A;\alpha_P]J \qquad \Gamma \vdash [\alpha_C]J}{\Gamma \vdash [\alpha_C;(\alpha_R;\alpha_A;\alpha_P)]J}$$

Once we've figured out the interesting control+physics we could go back and figure out the collision detection (in general though, it's better to try the easy cases first for the KeYmaera X proof. The opposite is sometimes true when figuring out a proof on paper, since we want to spend our energy on the interesting part). It's utterly essential that we don't keep the context Γ for the $J \vdash [\alpha_R;\alpha_A;\alpha_P]J$ branch, not even constant formulas, because in general $\Gamma_{\texttt{const}} \not\vdash [\alpha]\Gamma_{\texttt{const}}$ for hybrid games. That will *often* be true, but when it is it requires a proof. Here the rule HM is as follows:

$$\text{HM}\,\frac{\psi \vdash [\beta]\phi \qquad \Gamma \vdash [\alpha]\psi,\Delta}{\Gamma \vdash [\alpha;\beta]\phi,\Delta}$$

Whenever we use a hybrid systems rule with games for the first time, we should consider whether it will still be sound for games. Recall that some axioms and rules like V and G are not sound for games, but usually there's a related rule that is sound for games, in this case MR:

$$\text{MR}\,\frac{P \vdash Q}{[\alpha]P \vdash [\alpha]Q}$$

We can use MR to implement HM by letting $P = \psi, Q = [\beta]\phi$. This gives us a conclusion of $[\alpha][\beta]\phi$ which can then be rewritten as $[\alpha;\beta]\phi$ using the $[;]$ axiom.

Control Cases Instead we want to split into different proof cases for the different control cases. If you're doing the proof for the first time that's all you need to do, but I've actually looked into the future and seen the rest of the proof, so I can pre-empt a lot of the work. We've got a lot of control cases, and for each control case we'll also have to deal with physics. It would be nice if we can deal with physics once and for all. Luckily we can: all we'll need to prove the physics case is the invariant J which will still be true when physics start. So we can cut in $\phi \equiv (J \rightarrow [\alpha_P]J$ once now and use it when it comes up later. It's important that we cut it in early, before we branch, because then we only ever need to prove it once. [1] The cut here has an asterisk because we haven't proved it yet. Don't worry we'll come back and prove it. Also, these sequent proofs already take up a lot of space as it is, so we won't write out the assumption ϕ everywhere.

[1] When you can't see ahead into the future, a good alternative is to write a tactic that proves the cut each time it's needed, then use that tactic every time you see the physics case show up. It'll take more time for KeYmaera X, but still less time for you.

$$[\cup]\ \frac{\dfrac{}{\Gamma_{\texttt{const}}, J, \phi \vdash [?(x_2 > x_1); a_1 := A_1;][\alpha_A^d][\alpha_P]J}\ \text{Move Case} \quad \dfrac{}{\Gamma_{\texttt{const}}, J, \phi \vdash [?(x_2 - x_1 \leq d_{hit}); v_2 := P_1; a_2 := 0][\alpha_A^d][\alpha_P]J}\ \text{Punch Case} \quad \dfrac{}{\Gamma_{\texttt{const}}, J, \phi \vdash [\alpha_A^d][\alpha_P]J}\ \text{Do-Nothing Case}}{\text{cut}^* \dfrac{\Gamma_{\texttt{const}}, J, \phi \vdash [\alpha_R; \alpha_A^d; \alpha_P]J}{\Gamma_{\texttt{const}}, J \vdash [\alpha_R; \alpha_A^d; \alpha_P]J}}$$

The case where we both punch: Note we need to remember the fact that the test passed for Angel and use it to show that the same test passes for Demon. Also note that once we start assigning to variables we'll lose some, but not necessarily all of the assumptions in J. For typographic convenience, I'll write J^* to mean "whatever parts of J are still true".

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\ldots, x_2 - x_1 \leq d_{hit} \vdash x_2 - x_1 \leq d_{hit}}\ *}{\Gamma_{\texttt{const}}, J^*, x_2 - x_1 \leq d_{hit} \vdash \langle ?(x_2 - x_1 \leq d_{hit})\rangle\langle v_1 := -P_2; a_1 := 0\rangle[\alpha_P]J}\ \langle ?\rangle\text{id}}{\Gamma_{\texttt{const}}, J^*, x_2 - x_1 \leq d_{hit}, v_2 = P_1, a_2 = 0 \vdash \langle ?(x_2 - x_1 \leq d_{hit})\rangle\langle v_1 := -P_2; a_1 := 0\rangle[\alpha_P]J}\ [:=]}{\Gamma_{\texttt{const}}, J^*, x_2 - x_1 \leq d_{hit}, v_2 = P_1, a_2 = 0 \vdash \langle ?(x_2 - x_1 \leq d_{hit}); v_1 := -P_2; a_1 := 0\rangle[\alpha_P]J}\ \langle\cup\rangle}{\Gamma_{\texttt{const}}, J^*, x_2 - x_1 \leq d_{hit}, v_2 = P_1, a_2 = 0 \vdash \langle\alpha_A^d\rangle[\alpha_P]J}\ [d]}{\Gamma_{\texttt{const}}, J^*, x_2 - x_1 \leq d_{hit}, v_2 = P_1, a_2 = 0 \vdash [\alpha_A^d][\alpha_P]J}\ [:=]}{\Gamma_{\texttt{const}}, J, x_2 - x_1 \leq d_{hit} \vdash [v_2 := P_1; a_2 := 0][\alpha_A^d][\alpha_P]J}\ [?]}{\Gamma_{\texttt{const}}, J \vdash [?(x_2 - x_1 \leq d_{hit})][v_2 := P_1; a_2 := 0][\alpha_A^d][\alpha_P]J}\ [;]}{\Gamma_{\texttt{const}}, J \vdash [?(x_2 - x_1 \leq d_{hit}); v_2 := P_1; a_2 := 0][\alpha_A^d][\alpha_P]J}$$

The case where we both move:

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma_{\texttt{const}}, J^*, x_2 > x_1, a_1 = A_1 \vdash x_2 > x_1}\ *}{\Gamma_{\texttt{const}}, J^*, x_2 > x_1, a_1 = A_1 \vdash \langle ?(x_2 > x_1)\rangle\langle a_2 := -A_2\rangle[\alpha_P]J}\ \langle ?\rangle\text{id}}{\Gamma_{\texttt{const}}, J^*, x_2 > x_1, a_1 = A_1 \vdash \langle ?(x_2 > x_1); a_2 := -A_2\rangle[\alpha_P]J}\ \langle;\rangle}{\Gamma_{\texttt{const}}, J^*, x_2 > x_1, a_1 = A_1 \vdash \langle\alpha_A\rangle[\alpha_P]J}\ \langle\cup\rangle}{\Gamma_{\texttt{const}}, J^*, x_2 > x_1, a_1 = A_1 \vdash [\alpha_A^d][\alpha_P]J}\ [d]}{\Gamma_{\texttt{const}}, J, x_2 > x_1 \vdash [a_1 := A_1;][\alpha_A^d][\alpha_P]J}\ [:=]}{\Gamma_{\texttt{const}}, J \vdash [?(x_2 > x_1)][a_1 := A_1;][\alpha_A^d][\alpha_P]J}\ [?]}{\Gamma_{\texttt{const}}, J \vdash [?(x_2 > x_1); a_1 := A_1;][\alpha_A^d][\alpha_P]J}\ [;]$$

$$\text{cut}\ \frac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma_{\texttt{const}}, J^*, x_2 > x_1, a_1 = A_1, a_2 := -A_1 \vdash [\alpha_P]J}\ \text{CE},\mathbb{R}}{\Gamma_{\texttt{const}}, J^*, x_2 > x_1, a_1 = A_1, a_2 := -A_2 \vdash [\alpha_P]J}\ \langle:=\rangle}{\Gamma_{\texttt{const}}, J^*, x_2 > x_1, a_1 = A_1 \vdash \langle a_2 := -A_2\rangle[\alpha_P]J}}{\dfrac{}{J^*, \ldots \vdash J}\ \mathbb{R}} \quad \dfrac{}{J \vdash [\alpha_P]J}\ *(\text{by } \phi \in \Gamma)}$$

The case where we both do nothing:

$$\langle ?(true)\rangle \frac{\dfrac{*\ (\text{by } \phi \in \Gamma)}{J \vdash [\alpha_P]J}}{\langle\cup\rangle \dfrac{\Gamma_{\texttt{const}}, J \vdash \langle ?true\rangle[\alpha_P]J}{[d]\ \dfrac{\Gamma_{\texttt{const}}, J \vdash \langle\alpha_A\rangle[\alpha_P]J}{\Gamma_{\texttt{const}}, J \vdash [\alpha_A^d][\alpha_P]J}}}$$

Using a nice cut, we can reduce all three branches to the following case. Recall the definition of α_P:

$$\alpha_P \equiv \{x_1' = v_1, v_1' = a_1, x_2' = v_2, v_2' = a_2 \& x_1 <= x_2\}$$

The final case goes by dC and dI:

$$\begin{array}{c}\text{dI}\ \dfrac{*}{J, t = 0 \vdash [\alpha_P]v_1 + v_2 = 0} \quad \text{dI}\ \dfrac{*}{J, t = 0 \vdash [\alpha_P \& v_1 + v_2 = 0]J} \\ \text{dC}\ \dfrac{}{J \vdash [\alpha_P]J}\end{array}$$

3 Convergence

We already know how to prove loop properties inside a box: use induction, so we know that our favorite property holds no longer how long the loop runs. We've never discussed what to do with diamond stars, though. The difference now is that we have the freedom to choose how long the loop runs, so intuitively our job should be to decide how long we want to run the loop and prove if we actually run the loop that long, our postcondition will hold at the end. However, we might not be able to compute *exactly* how long the loop will have to run in advance, especially if our opponent is doing sneaky stuff every iteration. Instead of picking the exact number of iterations n in advance, we show the *existence* of such an n using what we call *convergence reasoning*.

Convergence can be understood in relation to the idea of termination metrics. If we had a really complicated `while` loop in C and wanted to prove that it terminates, what could we do? A very standard approach is to define a termination metric term θ carefully chosen so that (a) it is guaranteed to decrease (by at least some constant) every iteration, and (b) when the metric reaches zero, the guard condition will be false and the loop will end. Since the metric keeps getting smaller, it will eventually reach zero thus we've proven the loop terminates.

Convergence reasoning is intuitively similar, but the story is different for us because we don't have to worry whether a loop terminates. It will always terminate, it's just up to us how long we choose to run it before it terminates. Instead we're proving that *if* we run it long enough, then *when* we stop running it some nice desirable property of our choosing will be true! We do this by defining and proving a *convergence predicate* $\varphi(n)$, instead, where $\varphi(n)$

should be understood as "I can make the postcondition true after n more iterations". We could also think of this predicate as being $\varphi(\theta)$, where θ is now a *convergence term*: the argument has to φ has to get smaller every iteration until it reaches 0. The idea here is that we'll try to prove a convergence predicate $\varphi(n)$, Once we've chosen our property $\varphi(n)$ there are three more steps to the convergence argument:

1. Initially, it has to be possible to get into our happy state so long as we run the loop long enough. But we don't have to know *how long exactly* the loop will run. Given the meaning of φ, this comes out to $\Gamma \vdash \exists v.\varphi(v)$
2. The argument n has to get smaller each iteration, for the same reason termination metrics have to get smaller. This way we can keep running the loop and eventually we'll get n down to 0 (even if we don't know exactly how long yet). That is, $\vdash \forall v \geq 0.\varphi(v) \to \langle\alpha\rangle\varphi(v-1)$[2]
3. Our definition of $\varphi(n)$ tells us *when* we want to stop and the above case told us we will indeed stop eventually. Now we need to know that when we stop, we've actually stopped in a state where our desired postcondition is true. Since n doesn't actually have to be an integer, then subtracting 1 over and over again might not get us to exactly 0, just some arbitrary $n \leq 0$. If ϕ is our postcondition, we need to show $\exists v \leq 0.\varphi(v) \vdash \phi$.

Putting this all together we get a rule that rules:

$$\text{con'}\ \frac{\Gamma \vdash \exists v.\varphi(v) \qquad \vdash \forall v \geq 0.\varphi(v) \to \langle\alpha\rangle\varphi(v-1) \qquad \exists v \leq 0.\varphi(v) \vdash \phi}{\Gamma \vdash \langle\alpha^*\rangle\phi, \Delta}$$

Let's start with the simple example from lecture.

$$x > 0 \vdash \langle(x := x-1)^*\rangle x < 1$$

Notice how we want to get x to be smaller than 1 specifically. It helps us to think backwards form the third branch. Before we even choose a predicate φ let's think what the argument n a.k.a.θ should be. What's getting smaller each iteration, preferably smaller by 1? Well how convenient x is getting smaller by 1 and x shows up in the postcondition, so I bet we're gonna be able to use it as the convergence term. Now we can pick the convergence predicate $\varphi(v)$. At the end of the loop we get to know $\varphi(v)$ for some $v \leq 0$. Wait a second, if we have $x \leq 0$ then obviously $x < 1$! So let's just define $\varphi(v) \equiv (v = x)$.

[2]Why does n get smaller by exactly 1? It doesn't have to be that way, but decreasing n by your favorite constant $c > 0$ is not too hard to implement once you have this rule. And if you wanted to decrease it by a non-constant amount each iteration, that actually wouldn't work in general due to Zeno's paradox: you might never actually hit 0.

$$\text{con'}\,\frac{\mathbb{R}\,\dfrac{*}{\vdash \exists v.x = v} \qquad \forall_r\,\dfrac{\to_r\,\dfrac{\langle:=\rangle\,\dfrac{\mathbb{R}\,\dfrac{*}{v \geq 0, x = v \vdash x - 1 = v - 1}}{v \geq 0, x = v \vdash \langle x := x - 1\rangle x = v - 1}}{v \geq 0 \vdash x = v \to \langle x := x - 1\rangle x = v - 1}}{\vdash \forall v \geq 0. x = v \to \langle x := x - 1\rangle x = v - 1} \qquad \text{QE}\,\dfrac{*}{\exists v \leq 0. x = v \vdash x < 1}}{\vdash \langle (x := x - 1)^*\rangle x < 1}$$

That worked out! Aren't we glad we have QE!

4 Better, Faster, Stronger: Strategy for Angel

It should make intuitive sense that if Angel is *both* faster and stronger than Demon then he should have a winning strategy for knocking Demon out of the ring. In general, strategies for Angel loops can be tough to prove, so for the sake of this proof we're willing to assume Angel is *way* faster and stronger than Demon if that's what it takes to get through the proof. The easiest approach, though, is to try to prove that Angel wins and see figure out how strong we want our assumptions to be as we go. But also, before we go for a proof or even a convergence property and convergence term, we should really just work out at a high level what the strategy should even be anyway.

Strategy For simplicity let's assume we both start out exactly in the center of the ring. This makes Angel's first turn easy to decide because he can't move forward, and staying still is boring. The only interesting action is to punch Demon. Then Demon can respond by either punching back, doing nothing, or braking. Let's look at the result in each case:

- Demon punches back: In this case both of us are now accelerating toward the edges l and r and neither of us are braking so we're not really doing anything to stop it. How much does this help/hurt the players? That's totally a question of how long the ODE runs. Speaking of which, how long should the ODE run and which player should even get to pick?
  - Demon picks? That won't work at all because Demon could always pick duration 0 and trivially filibuster us.
  - Angel picks? This is really giving Angel an upper hand, but if Demon were smart enough we could imagine him still having a winning strategy. Since this is going to be a hard proof either way, giving Angel an upper hand is honestly not a bad idea right now.

– Nobody picks? If we use a test to take all (or even some!) of the nondeterminism out of the ODE, we could take away Angel's big advantage without making the game a trivial win for Demon. This is an interesting possibility, but let's not go there yet because giving Angel an advantage is convenient right now.

Ok so Angel is going to pick the ODE duration. If we're at the center and Demon is moving faster than Angel (because Angel punches harder), then Demon is going to fall off before Angel and Angel will win if he picks the right ODE duration. So if Demon punches us back we have a winning strategy, nice!

- Demon applies the brakes: Angel can still respond by picking his favorite duration for the ODE, which gives us a lot of power. But running the ODE as long as possible doesn't help anymore: If we run the ODE too long then we're not going to knock Demon off the other side of the board. Eventually if Demon accelerates long enough he'll get right back to where he started and we won't go anywhere at all. We could keep hitting him back but this would just be really roundabout filibuster for Demon.

  Some intuition from fighting games is helpful here. If you're trying to get the opponent to the other side of the arena, you do two things: *first* you hit them, *then* you run to catch up with them. The reason you do this is because hitting them *once* won't be enough to win the game. You need to repeatedly get within hitting distance and then pummel them until you win. What makes this a serious interesting strategy is that our strategy plays out over *multiple iterations* of the loop: half the time we're punching the enemy, half the time we're catching up with them.

- Demon does nothing: This is just an easier case of punching: we can evolve the ODE as long as we want and knock Demon out of the arena.

To summarize, the winning strategy for Angel looks like this: At each step, check if you're in punching distance. Punch if you can, and if Demon punches back you win. If not, let the ODE run "a while" so they move to the right, but not too long. If you weren't within punching distance, it must be because this is the second turn of the "punch-and-catch-up" strategy. So run and catch up with them. If we picked all the ODE durations right, we should be able to prove that *when* you catch up, you've made progress against them.

Picking ODE Durations: Toward a Convergence Property At this point we've figured out the *discrete* aspect of our strategy: whether to punch vs. run. But this is a hybrid game, so we need to make continuous decisions in our strategy as well: how long to run the ODE. As hinted above, that strategy will be informed by physical reasoning. But seriously where do we even start with this part of the strategy?

Recall that we eventually want to do this proof with a convergence property. A convergence property is something that needs to keep getting better and better each time we run the loop

until its eventually so great that we just win the game. We win the game when $x_2 > r$, so the natural thing to say is that x_2 has to get bigger each time. Not only that but we're going to need to know *how much bigger* it gets each round. We need to avoid a Zeno's paradox situation where x_2 never actually reaches r and just gets super close. That's usually easy to avoid: we just need to compute how much progress we actually make in the worst case.

Second, note that it has to get better *each* round. There are two "kinds" of rounds in our strategy: punching rounds and catching-up rounds. We need to make progress in both kinds of rounds.

From a physics perspective, what would make that possible? It would make things *way* easier if we knew that Demon never ever moved left at all ever and was always moving to the right. We can make that happen by guaranteeing that even in the worst case where Demon is always braking, we catch up to him exactly when his velocity reaches 0. So the important time to consider is the stopping time T_s where Demon will be back to velocity 0. Since the initial velocity is P_1 (Angel's punch velocity) and the braking acceleration is A_2 (Demon acceleration), the stopping time is our favorite formula:

$$T_s = \frac{P_1}{A_2}$$

We need to split the T_s time between the punching phase and the catching-up phase (we don't want to catch up immediately, else Demon never actually moves). Unless we have a reason not to, let's do the simplest thing: spend $\frac{1}{2} \cdot T_s$ punching and the other $\frac{1}{2} \cdot T_s$ time catching up. Now we should try to compute how far Demon moves in each phase of the strategy and that will give us a bound we can use in the convergence property.

How far do we move during the punching phase?: If we're punching and Demon is braking, then Angel is going to be stationary and Demon is going to follow the familiar parabolic trajectory:

$$x_{2,mid}(t) = x_{2,init} + P_1 \cdot t + A_2 \frac{t^2}{2}$$

so when we plug in $t = \frac{P_1}{2 \cdot A_2}$ we get $x_{2,init} + \frac{P_1^2}{2 \cdot A_2} - \frac{P_1^2}{8 \cdot A_2} = x_{2,init} + \frac{3 \cdot P_1^2}{8 \cdot A_2}$. This should make some intuitive sense: we'll be moving a total of $\frac{P_1^2}{2 \cdot A_2}$ distance and we would expect most of that to happen toward the beginning when we're still moving fast.

How far do we move during the braking phase? By the argument above we ought to be moving $\frac{P_1^2}{8 \cdot A_2}$ during this part, and the math confirms it:

$$x_{2,final}(t) = x_{2,mid} + (P_1 - A_2 \cdot T_s/2) \cdot T_s/2 + A_2 \frac{t^2}{2}$$

Plugging in $t = T_S$

$$\begin{aligned} x_{2,final}(t) =& x_{2,mid} + (P_1 - (P_1/2)) \cdot (P_1/A_2)/2 - \frac{P_1^2}{8 \cdot A_2} \\ =& x_{2,mid} + 1/4 \cdot (P_1^2/A_2) - \frac{P_1^2}{8 \cdot A_2} \\ =& x_{2,mid} + 1/8 \cdot (P_1^2/A_2) \end{aligned}$$

We also want Angel to move the same distance during the same time, so set $A_1 \cdot (T_S/2)^2/2 = P_1^2/A_2^2/2$ and solve for A_1. $A_1 \cdot (T_S/2)^2/2 = A_1(P_1/A_2)^2/8 = A_2(P_1/A_2)^2/2$

so $A_1 \geq 4 \cdot A_2$: If we can accelerate four times as hard as Demon we'll be able to win. I guess let's put that in our precondition. Maybe that's a bit of steep requirement for real life, but this is just our first proof. You can do much better if you try harder.

Picking a Convergence Property Based on our analysis, we get Demon to move at least $1/8 \cdot (P_1^2/A_2)$ every iteration of the loop. We're almost ready to write a convergence property, but we need to beware the following:

- Does this actually converge? To converge, we need $1/8 \cdot (P_1^2/A_2)$ to be constant or increasing, otherwise we could get into a Zeno paradox. Thankfully this is a constant expression (because P_1 and A_2 are constants), so we're solid.
- What else needs to go in our convergence property $\varphi(v)$? Recall that the inductive step assumes $\varphi(v)$ and proves $\varphi(v-1)$, so any assumptions we need in the inductive step have to be part of $\varphi(v)$. In that sense picking a convergence predicate is a lot like picking an invariant, except that $\varphi(v)$ varies with v, which is why we sometimes call it a *variant*. Note that even though $\varphi(v)$ varies with v, we can also include formulas that talk about all sorts of other variables in $\varphi(v)$, so long as those extra formulas are invariant. For example $A_1 \geq 4 \cdot A_2$ is pretty important to our proof so we'll want to put it in the variant even though it's not directly related to v.

The super essential part of our variant argument is that we get closer to r, so we could start with:

$$\varphi'(v) \equiv x_2 + v \cdot \frac{P_1^2}{8 \cdot A_2} \geq r$$

By our super strategic argument above, we ought to be able to decrease v by 1 each time we run the loop: x_2 keeps getting further to the right. It's certainly true initially: it holds true no matter where x_2 is, just with different values of v: v says how long we need to play our strategy.

The main other thing to look out for is the invariant part of our variant. This whole time we've been assuming that we're moving to the right, so we should put $v_1 \geq 0 \wedge v_2 \geq 0$ in the invariant. We've also been assuming that we never go left of the center point (relevant for the case where both players punch) and that we're always to the left of Demon. Plus we've got all our constant assumptions about constants like l, r, A_1, A_2, P_1, P_2. So something like this is a good variant.

$$\varphi(v) \equiv x_2 + v \cdot \frac{P_1^2}{8 \cdot A_2} \geq r \wedge l < r \wedge v_1 \geq 0 \wedge v_2 \geq 0 \wedge x_1 \leq x_2 \wedge A_1 \geq 4 \cdot A_2 \wedge P_1 \geq P_2 \wedge A_2 > 0 \wedge P_2 > 0$$

That's a mouthful, but nobody ever said this would be easy.

Do the Proof Now we're proving the following theorem:

$$\begin{aligned} Pre_W &\equiv P_1 \geq P_2 \wedge A_1 \geq 4 \cdot A_2 \\ \Gamma_{\texttt{const}} &\equiv l < r \wedge A_1, A_2, P_1, P_2 > 0 \\ Pre_G &\equiv \Gamma_{\texttt{const}} \wedge v_1 = v_2 = 0 \wedge x_1 = x_2 = (l + r)/2 \\ Pre &\equiv Pre_W \wedge Pre_G \\ Post &\equiv x_2 > r \\ Thm &\equiv Pre \rightarrow \langle \alpha \rangle Post \end{aligned}$$

We start this time with the convergence rule, using our convergence property from above. The precondition and postcondition prove easily for Mathematica even if they weren't obvious to us. The inductive case is the interesting part of the proof, but even in that branch there are a few propositional rules we can easily get out of the way before continuing the proof later

$$\text{con}' \frac{\mathbb{R} \dfrac{*}{Pre \vdash \exists v.\varphi(v)} \quad \mathbb{R} \dfrac{*}{\exists v \leq 0.\varphi(v) \vdash \phi} \quad \forall\text{R} \dfrac{\rightarrow\text{R} \dfrac{\rightarrow\text{R} \dfrac{v \geq 0, \varphi(v) \vdash \langle(\alpha_R; \alpha_A^d; \alpha_P)\rangle \varphi(v-1)}{v \geq 0 \vdash \varphi(v) \rightarrow \langle(\alpha_R; \alpha_A^d; \alpha_P)\rangle \varphi(v-1)}}{\vdash v \geq 0 \rightarrow \varphi(v) \rightarrow \langle(\alpha_R; \alpha_A^d; \alpha_P)\rangle \varphi(v-1)}}{\vdash \forall v.v \geq 0 \rightarrow \varphi(v) \rightarrow \langle(\alpha_R; \alpha_A^d; \alpha_P)\rangle \varphi(v-1)}}{Pre \vdash \langle(\alpha_R; \alpha_A^d; \alpha_P)^*\rangle (x_2 > r)}$$

Case Analysis At this point we want to apply some strategy. Strategic decisions will involve inspecting the state to determine what's most useful right now. In our case we want to punch, but that's only useful if we're close to Demon. Otherwise it's much more useful to move. We can dream up a simple case-analysis rule that allows us to case on truth/falsehood of formulas. [3]

[3] This drives constructivists nuts, but we're in a classical logic so HAH!

$$\text{case}(P)\,\frac{\Gamma, P \vdash \Delta \qquad \Gamma, \neg P \vdash \Delta}{\Gamma \vdash \Delta}$$

Any time we invent a rule we need to justify it. The case analysis rule is actually a derived rule, meaning we can prove it using existing rules:

$$\text{cut}\,\frac{\vee\text{L}\,\dfrac{\Gamma, P \vdash \Delta \qquad \Gamma, \neg P \vdash \Delta}{\Gamma, (P \vee \neg P) \vdash \Delta} \qquad \text{hide}\,\dfrac{\mathbb{R}\,\dfrac{*}{\vdash (P \vee \neg P)}}{\Gamma \vdash (P \vee \neg P), \Delta}}{\Gamma \vdash \Delta}$$

We use the case rule to check a formula P (mnemonic: "I can **P**unch Demon right now") where $P \equiv (x_2 - x_1 \leq d_{hit})$:

$$\text{case}\,\frac{v \geq 0, \varphi(v), P \vdash \langle(\alpha_R; \alpha_A^d; \alpha_P)\rangle\varphi(v-1) \qquad v \geq 0, \varphi(v), \neg P \vdash \langle(\alpha_R; \alpha_A^d; \alpha_P)\rangle\varphi(v-1)}{v \geq 0, \varphi(v) \vdash \langle(\alpha_R; \alpha_A^d; \alpha_P)\rangle\varphi(v-1)}$$

Punching Case We start with the case where we can actually punch Demon. We use the assumption P to pick the right case of our controller, then look at the different reactions.

$$\begin{array}{r}
[\cup]\,\dfrac{\text{Three cases}}{v \geq 0, \varphi(v), P, v_2 = P_1 \vdash [\alpha_A]\langle\alpha_P\rangle\varphi(v-1)} \\
\langle d\rangle\,\dfrac{}{v \geq 0, \varphi(v), P, v_2 = P_1 \vdash \langle\alpha_A^d\rangle\langle\alpha_P\rangle\varphi(v-1)} \\
\langle;\rangle\,\dfrac{}{v \geq 0, \varphi(v), P, v_2 = P_1 \vdash \langle\alpha_A^d; \alpha_P\rangle\varphi(v-1)} \\
\langle:=\rangle,\text{id}\,\dfrac{}{v \geq 0, \varphi(v), P \vdash \langle v_2 := P_1\rangle\langle\alpha_A^d; \alpha_P\rangle\varphi(v-1)} \\
\langle?\rangle,\text{id}\,\dfrac{}{v \geq 0, \varphi(v), P \vdash \langle?(P)\rangle\langle v_2 := P_1\rangle\langle\alpha_A^d; \alpha_P\rangle\varphi(v-1)} \\
\langle;\rangle\,\dfrac{}{v \geq 0, \varphi(v), P \vdash \langle?(P); v_2 := P_1;\rangle\langle\alpha_A^d; \alpha_P\rangle\varphi(v-1)} \\
\langle\cup\rangle\,\dfrac{}{v \geq 0, \varphi(v), P \vdash \langle\alpha_R\rangle\langle\alpha_A^d; \alpha_P\rangle\varphi(v-1)} \\
\langle;\rangle\,\dfrac{}{v \geq 0, \varphi(v), P \vdash \langle(\alpha_R; \alpha_A^d; \alpha_P)\rangle\varphi(v-1)}
\end{array}$$

We split into three cases based on Demon's reaction. We start with the easy cases.

Punch-Do-Nothing Case: This is a special case of the Punch-Punch case where the second punch velocity is 0, so we're going to skip it.

Punch-Punch Case: In the punch case we update velocities, but then we have to check whether we've collided. Since we're punching away from each other, we didn't collide. But we do get two cases so we cut in a case to handle them both. The case we cut in is:

$$v \geq 0, \varphi(v), P, v_2 = P_1, v_1 = -P_2 \vdash \langle ODE \rangle \varphi(v-1)$$

$$\alpha_C \equiv \{?(x_1 = x_2 \wedge v_1 >= v_2); v_1 := 0; v_2 := 0; a_1 := 0; a_2 := 0; \} \cup \{?(x_1 \neq x_2 \vee v_1 < v_2); \}$$
$$\alpha_P \equiv \alpha_C; \alpha'_P$$

$$\cfrac{\cfrac{\cfrac{\cfrac{\text{To be proved}}{v \geq 0, \varphi(v), P, v_2 = P_1, v_1 = -P_2 \vdash \langle \alpha_P \rangle \varphi(v-1)}}{v \geq 0, \varphi(v), P, v_2 = P_1 \vdash [v_1 := -P_2] \langle \alpha_P \rangle \varphi(v-1)}}{v \geq 0, \varphi(v), P, v_2 = P_1 \vdash [?(P)][v_1 := -P_2] \langle \alpha_P \rangle \varphi(v-1)}}{v \geq 0, \varphi(v), P, v_2 = P_1 \vdash [?(P); v_1 := -P_2] \langle \alpha_P \rangle \varphi(v-1)}$$

To prove the cut we'll need to use the diamond rule for ODEs. We use a very simple version because we can get away with it:

$$\langle' \rangle \frac{\exists t. [x := y(t)] P}{\langle x' = \theta \rangle P}$$

We then plug in $t = (r - x_2)/P_1$ since that's how long it takes to kill Demon:

$$[:=], \mathbb{R} \; \cfrac{*}{\exists \mathrm{R} \; \cfrac{v \geq 0, \varphi(v), P, v_2 = P_1, v_1 = -P_2, t = (r-x_2)/P_1 \vdash [x_1 := x_1 + P_2 \cdot t; x_2 := x_2 + P_1 \cdot t;] \varphi(v-1)}{\langle' \rangle \; \cfrac{v \geq 0, \varphi(v), P, v_2 = P_1, v_1 = -P_2 \vdash \exists t. [x_1 := x_1 + P_2 \cdot t; x_2 := x_2 + P_1 \cdot t;] \varphi(v-1)}{v \geq 0, \varphi(v), P, v_2 = P_1, v_1 = -P_2 \vdash \langle ODE \rangle \varphi(v-1)}}$$

Punch-Move case: In our strategy above, in the punch case we're always at exactly the same point. We forgot to add that to our φ before so let's add it now, it's actually super important!

$$\varphi(v) \texttt{+=} \text{“}(P \rightarrow x_1 = x_2)''$$

As before, we solve the ODE and get a big program that inputs the solutions:

$$\alpha_{sol} \equiv \left\{ v_2 := P_1 + A_2 \cdot t; x_2 := x_2 + P_1 \cdot t - A_2 \frac{t^2}{2} \right\}$$

We plug in the time $T_s/2$ from our physics solution, and QE should be able to tell us that the convergence property is satisfied:

$$\mathbb{R}\,\exists\mathrm{L}\,\langle'\rangle\,[:=]\,\frac{*}{\begin{array}{c}v\geq 0,\varphi(v),P,v_2=P_1,a_2=-A_2,t=T_s/2\vdash[\alpha_{sol}]\langle\alpha_P\rangle\varphi(v-1)\\ \hline v\geq 0,\varphi(v),P,v_2=P_1,a_2=-A_2\vdash\exists t.[\alpha_{sol}]\langle\alpha_P\rangle\varphi(v-1)\\ \hline v\geq 0,\varphi(v),P,v_2=P_1,a_2=-A_2\vdash\langle\alpha_P\rangle\varphi(v-1)\\ \hline v\geq 0,\varphi(v),P,v_2=P_1;\vdash[a_2:=-A_2;]\langle\alpha_P\rangle\varphi(v-1)\end{array}}$$

Move-Punch case: That covered all the cases where we punch Demon. The only other thing we'll ever do is accelerate forward. In this case, Demon might attempt to punch us, but that will blow up by contradiction because we only move when Demon is too far away to punch us successfully:

$$\mathrm{hide}\,[?]\,[;]\,\frac{\mathbb{R}\,\dfrac{*}{\neg P,P\vdash}}{\begin{array}{c}v\geq 0,\varphi(v),\neg P,P\vdash[v_1:=-P_2]\langle\alpha_P\rangle\varphi(v-1)\\ \hline v\geq 0,\varphi(v),\neg P\vdash[?(P)][v_1:=-P_2]\langle\alpha_P\rangle\varphi(v-1)\\ \hline v\geq 0,\varphi(v),\neg P\vdash[?(P);v_1:=-P_2]\langle\alpha_P\rangle\varphi(v-1)\end{array}}$$

Move-Move case: In the move-move case we get accelerations $A_1,-A_2$ respectively. This is the "catch-up" case. As in the discussion above, we evolve for time $T_S/2$ and at the end we'll be all caught up to Demon. When we catch up, we will have moved far enough to get $\varphi(v-1)$ and QE takes care of the rest.

$$\mathbb{R}\,\exists\mathrm{L}\,\langle'\rangle\,\frac{*}{\begin{array}{c}v\geq 0,\varphi(v),\neg P,a_1=A_1,a_2=-A_2,t=T_S/2\vdash[\alpha_{sol}]\langle\alpha_P\rangle\varphi(v-1)\\ \hline v\geq 0,\varphi(v),\neg P,a_1=A_1,a_2=-A_2\vdash\exists t.[\alpha_{sol}]\langle\alpha_P\rangle\varphi(v-1)\\ \hline v\geq 0,\varphi(v),\neg P,a_1=A_1,a_2=-A_2\vdash\langle\alpha_P\rangle\varphi(v-1)\end{array}}$$

Well that would be great if it were true, except if you did this proof in KeYmaeraX you would find out that I lied. (Don't freak out, it's easy enough to fix, and I won't make you go through everything again) We were actually missing something from $\varphi(v)$ all along: all we remembered was that both players are moving to the right, we never remembered anything about their relative velocities. For the move-move case all we controlled was the acceleration, so it actually does matter a lot that we understand the velocities. We want to say something like they'll have the same velocity and position at time $T_s/2$, so I guess we need something in the invariant like $x_1\neq x_2\rightarrow(v_1+a_1\cdot T_s/2=v_2+a_2\cdot T_s/2\wedge x_1+v_1\cdot T_s/2+a_1\cdot T_s^2/8=x_2+v_2\cdot T_s/2+a_2\cdot T_s^2/8)$, where $x_1\neq x_2$ is how we know we're in the "move" part of the strategy.

That doesn't quite finish the story though. In the Punch-Move case we assumed Angel was stationary, and after a round of punching-and-catching up he won't be. The saving grace

is the collision detection! Our punching case assumes our positions will be exactly equal, something which we can easily ensure as an invariant in the move-move case. Now, next time we get to the punching case, the collision logic will fire, restoring our velocities and accelerations to 0. We don't need to express anything about collision in the convergence property though, that'll just be part of the proof for the punching case. Phew! After much blood, sweat and tears we've fixed our proof.

5 Extra: Yet Another con' example

Editor's Note: In general we've been making a big push to use models of realistic example systems in the recitations. The cost is that they are much more complicated, both to model and especially to prove, than toy examples. Especially for advanced topics like games, there's the risk that you won't get the advanced examples as easily on the first try. So here is an additional smaller, simpler, but less fun example from an old version of the recitation, should you wish for a second not-too-crazy example.

So now that we've got a full understanding of the simpler proof under our belts, let's try using the con' rule to prove a more complex game.

$$\vdash \langle x := 0; (x' = -1)^d; (x := x + 1)^* \rangle x > 0$$

So Angel lets Demon make x as small as he wants to. He maybe even goes make a cup of tea while this is happening - who knows! Anyway, when he comes back, x is really super small. The good news is that Angel is in heaven, which is really one big permanent holiday, so he gets to keep adding 1 to x, however many times he wants. And he wants to make sure that x becomes positive again, not negative like before.

We can use $x > -v$ as our convergence property, and start with decomposing our problem.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{t \geq 0 \vdash [x := -t]\langle (x := x + 1)^* \rangle x > 0}\textit{ to be continued...}}{t \geq 0 \vdash [x := 0][x := x - t]\langle (x := x + 1)^* \rangle x > 0}[:=]}{\vdash [x := 0]\forall t \geq 0.[x := x - t]\langle (x := x + 1)^* \rangle x > 0}\forall_r, \to_r}{\vdash [x := 0][x' = -1]\langle (x := x + 1)^* \rangle x > 0}\text{ODE}}{\vdash [x := 0]\langle (x' = -1)^d \rangle \langle (x := x + 1)^* \rangle x > 0}\langle^d\rangle}{\vdash \langle x := 0 \rangle \langle (x' = -1)^d \rangle \langle (x := x + 1)^* \rangle x > 0}\langle:=\rangle}{\vdash \langle x := 0 \rangle \langle (x' = -1)^d; (x := x + 1)^* \rangle x > 0}\langle;\rangle}{\vdash \langle x := 0; (x' = -1)^d; (x := x + 1)^* \rangle x > 0}\langle;\rangle$$

There are a couple of cool things happening! First of all, notice how we applied some rules to inner formulas instead of only to the outer-most formula! We are allowed to do that because

we used axioms instead of proof rules (which only go one way). Second, we get to have fun carrying a delayed assignment around! *YAY!*

Where do we carry the delayed assignment to though? Every branch? The key issue here is that the assignment only applies to the original variables, i.e. the ones that Γ mentioned. It gets passed around as context - if the context isn't there because of soundness, then neither should the delayed assignment!

$$\dfrac{\exists_r v \mapsto \text{-}t\text{-}1 \dfrac{\mathbb{R}\dfrac{*}{t \geq 0 \vdash -t > -t-1}}{\{:=\}\dfrac{t \geq 0 \vdash \exists v.-t > -v}{\text{con'}\dfrac{t \geq 0 \vdash \{x := -t\}\exists v.x > -v}{t \geq 0 \vdash \{x := -t\}\langle (x := x+1)^*\rangle x > 0}}} \quad \text{The big branch!} \quad \text{QE}\dfrac{*}{\exists v \leq 0.x > -v \vdash x > 0}}{t \geq 0 \vdash \{x := -t\}\langle (x := x+1)^*\rangle x > 0}$$

And finally, the last branch is very similar to the one from the simple example.

$$\dfrac{\dfrac{\dfrac{\mathbb{R}\dfrac{*}{v \geq 0, x > -v \vdash x+1 < -v+1}}{\langle :=\rangle \dfrac{v \geq 0, x > -v \vdash \langle x{+}{+}\rangle x > -v+1}{\rightarrow_r \dfrac{v \geq 0 \vdash x > -v \rightarrow \langle x{+}{+}\rangle x > -v+1}{\forall_r \dfrac{}{\vdash \forall v \geq 0.x > -v \rightarrow \langle x{+}{+}\rangle x > -v+1}}}}}$$

And thus is the proof concluded, and the Angel hath won this most epic Battle of the Tea & Holidays.

6 Extra: Avoiding Mistakes in Game Proofs

The nice thing about doing proofs in KeYmaera X is that it's always careful so we don't have to be. However, when we're doing paper proofs on the assignments, we need to be careful because there's no KeYmaera X to look out for us. Here's an example of *not* being careful. We're going to fail to prove the true property $x = 0 \vdash \langle (x := 0 \cup x := 1)^\times\rangle x = 0$ and we're gonna fail real hard:

$$\dfrac{\dfrac{\dfrac{*}{x = 0 \vdash 0 = 0}}{[:=]\dfrac{}{x = 0 \vdash [x := 0]x = 0}} \quad [:=]\dfrac{x = 0 \vdash 1 = 0}{x = 0 \vdash [x := 1]x = 0}}{[\cup]\dfrac{x = 0 \vdash [x := 0 \cup x := 1]x = 0}{\text{ind}\dfrac{x = 0 \vdash [(x := 0 \cup x := 1)^*]x = 0}{\langle^d\rangle\dfrac{x = 0 \vdash [(x := 0 \cup x := 1)^*]x = 0}{x = 0 \vdash \langle (x := 0 \cup x := 1)^\times\rangle x = 0}}}}$$

The end property says that Angel wants to get $x = 0$, and he can achieve that in each iteration of the loop, but Demon can decide to repeat. We saw that he isn't allowed to repeat indefinitely, so Angel should win.

Uhh... what did we do wrong? What happened is that when converted the Demon loop into an Angel loop (the ind step), we accidentally deleted some dual operators and wrote down the wrong formula! As you might imagine, if you write down the wrong formula you will fail to prove what you wanted! For this reason, we highly encourage you to rewrite all programs using *only* dual operators, and removing all Demon-specific notation like $\alpha \cap \beta$ and $\alpha^\times$.

In this case, we'd get:

$$(x := 0 \cup x := 1)^\times \equiv ((((x := 0)^d \cup (x := 1)^d)^d)^*)^d$$

Since assignment is deterministic, that reduces to:

$$(((x := 0 \cup x := 1)^d)^*)^d$$

So we had actually forgotten a few dual operators. Silly TA. Because we fully understand dual operators, the proof now has *no chance* of going wrong! *FACT!*

$$\langle^d\rangle \dfrac{\text{ind} \dfrac{[^d] \dfrac{\langle\cup\rangle \dfrac{[:=] \dfrac{*}{\vdash 0 = 0, \langle x := 1\rangle x = 0}}{\vdash \langle x := 0\rangle x = 0, \langle x := 1\rangle x = 0}}{\vdash \langle x := 0 \cup x := 1\rangle\, x = 0}}{\vdash \left[((x := 0 \cup x := 1)^d\right] x = 0}}{\vdash \left\langle \left(\left((x := 0 \cup x := 1)^d \right)^* \right)^d \right\rangle x = 0}$$