The Logic Game

Winter 2026

Goal: Pedagogical Cohesion (The "Logic Game"): Like Java for the programming stream, it is important for the students to have a common language across courses. Our students struggle to connect the abstract logic and theory with the intuition we want them to have about computer science concepts. The human subconscious has always understood complex things through stories and games. We propose a unified pedagogical framework—**The Logic Game**—where First Order Logic is used not by formally manipulating symbols, but as a dynamic game between a *Prover*, an *Adversary*, and an *Oracle*. This intuition allows students to "play" definitions rather than memorize the ideas taught in Discrete Math EECS 1019 and Theory of Computation EECS 2001. One instructor's little brother studying set theory at U of T asked him for help. Many students over the years have asked him for help with the epsilon-delta proofs they see in calculus. They are shown hard proofs without any understanding of where these proofs come from. A light goes off when they are shown this game. This instructor helps them express what they need to prove as a statement $\forall \epsilon, \exists \delta \ldots$ and then convert this into a game in which an adversary gives an epsilon and in response the prover returns a smaller delta. This in turn becomes the proof "Let ϵ be arbitrary. Let $\delta = \frac{1}{2}\epsilon$." In contrast, without intuition, the formal proof has a Universal Instantiation rule to remove the $\forall \epsilon$ and a Existential Instantiation rule to remove the $\exists \delta$.

"This approach is grounded in Game-Theoretical Semantics, introduced by Jaakko Hintikka (1968), which formally defines logical truth as the existence of a winning strategy for the verifier in a game against a falsifier."

References: [1] Hintikka, Jaakko. "Language-games for quantifiers." American Philosophical Quarterly Monograph Series 2 (1968): 46-72. [2] Hintikka, Jaakko. Logic, Language-Games and Information: Kantian Themes in the Philosophy of Logic. Clarendon Press, 1973.

`https://en.wikipedia.org/wiki/Game_semantics`

Pedagogical Motivation of the Logic Game:
To be taught in the first class.

- **Experience with the Method**: The proponent has taught this logic game in the beginning of his 1019, 1090, 2001, 2101, and 3101 courses for a few years. Even starting with a quite hard example, two thirds say they understand it the first day. He finds that once they get it, they really do love it and find it better explains almost all of the course material to them.

- **Text:** There is currently no textbook, but the instructor has thousands of slides which could be turned into a book.

- **Understanding Inward Out about Free Variables:** Because the truth of a statement depends on the values of its *free* variable, the statement is *about* these without mentioning the *bound* variables.
  - $A(I) = P(I) \equiv$ "Algorithm A correctly solves problem P on input I."
  - $\forall$ input I $[A(I) = P(I)] \equiv$ "A correctly computes P."
  - $\exists$ Algorithm A $[\forall$ input I $A(I) = P(I)] \equiv$ "P is computable".
  - Students need to know that $\forall$ input I $\exists$ Algorithm A $A(I) = P(I)$ is **wrong.**

- **Standard Proof Techniques:**

$A \rightarrow B$: Implication is proved assuming A and proving B. Let's tell the story that an oracle is assuring and supporting us to use A and that a prover (or us) needs to prove B.

Order of Game follows Parse Tree: $\forall x \exists y$ is read left to right.

Prove $\forall x$: Practical math papers do not handle quantifiers as currently taught in 1090. The standard thing is to says "Let x be arbitrary." But the students need to understand that this object is not chosen by them, or by the professor, or randomly. It is a worst case value chosen by an adversary who replies "You say something is true for all objects x. What about my object x'?"

Prove $\exists y$: The standard proof goes on to say "Let $y = x + 1$." The student needs to see that the prover, knowing the value from the adversary, proves the existence of such a y by constructing such an object. We might as well tell the story that the prover provides it.

Assure $\forall x$: The students also struggle with the fact that this game turns around when the oracle is assuring us of an axiom. When she assures us of that something is true for all objects x, we give her our favorite object x' and she assures us that it is true for that.

Assure $\exists y$: When she assures us of that there exists an object y with some property, she gives us such a y'.

Valid: A sentence is *valid/tautology* iff it is true in every universe in which the axioms are true. We *prove* this by providing a strategy in which the prover, with help from the oracle, can win this game against the adversary.

- **An Instructive Game:** We each choose an integer. If mine is bigger, I win. Because I am nice, I will let you go first. The students will laugh and see that the second player will win because he knows the first player's number. This relates to the logic $\forall x, \exists y, y > x$. Once students grasp this dynamic, we apply it to most theoretical concepts in the curriculum.

- **Replacing Details with Intuition:** Currently EECS 2001 gets the students to give detailed designs of Turing Machines for solving given toy problems. This may need to be eliminated to make room for other things. However, by studying our game the students can gain a deep understanding of the beauty in Turing's definition of a problem P being computable:

$$\exists \text{ TM } M, \forall \text{ input } I, M(I) = P(I).$$

Because Turing did not want to restrict which operations his machines could do, he allowed it to have any finitely described transition function. This allows the coder/prover to have his machine M be powerful enough to multiply two 1000 digit integers together in one time step. This does not faze the adversary, because knowing this M, he makes the input I much bigger than this.

- **Translation English to Understanding:** We want students to be able translate the English sentence "Java programs can be simulated by a TM" into the first order logic sentence

$$\forall \text{ Java } J, \exists \text{ TM } M, \forall \text{ input } I, M(I) = J(I),$$

and then into the understanding that if a *compiler* oracle is assuring us that this is true, then we can give her any Java program J, and she will return a TM M, such that for any input I given to us by an adversary, the two output the same answer. The students should be able to intuit that from this axiom it follows that the set of computational problems computable by a Java program is a subset of those computable by a TM, namely

$$\forall \text{ Problem } P, [\exists \text{ Java } J, \forall \text{ input } I, J(I) = P(I) \rightarrow \exists \text{ TM } M, \forall \text{ input } I, M(I) = P(I)]$$

The proposed course will teach how both the proof and its intuition follows easily from the game and that the game follows easily from the parsing of the logical sentence.

- **Game:**

Axiom: We are assured by *compiler* oracle the axiom $\exists$ TM $M, \forall$ input $I, M(I) = P(I)$.

Prove: $\forall$ Problem $P, [\exists$ Java $J, \forall$ input $I, J(I) = P(I) \rightarrow \exists$ TM $M, \forall$ input $I, M(I) = P(I)]$

$\forall$**P:** The adversary gives a worst case P.

$\rightarrow$**:** The *Java* oracle assures the LHS. The prover proves the RHS: $\exists$ TM $M, \forall$ input $I, M(I)=P(I)$

$\exists$**M:** The prover proves this by constructing a TM M.

$\exists$**J:** He does this by taking the Java program J given by the Java oracle.

$\forall$**J** $\exists$**M:** He gives this J to the compiler oracle, who gives back a TM M.

$\forall$**I:** The prover then continues to prove the RHS by taking a worst case input I from his adversary, which he passes on to both oracles.

=**:** The compiler oracle assures $M(I)=J(I)$, the Java oracle assures $J(I)=P(I)$, and he uses transitivity to prove his final statement $M(I)=P(I)$.

Proof by Construction: Because he constructed M with the required properties, he has proved the RHS.

Deduction: Because this was done with the assurance of the LHS, this proves the implication.

Universal Generalization: Because it was done for a worst case P, it has been done for all P.

Logical Consequence: Because it was proved with help from the axiom, it is true in every model/universe in which this axiom is true.

- **Standard Math Proof (With Formal Names):**

Axiom: Assume axiom.

$\forall$**P:** Let P' be arbitrary. (Arbitrary/Worst Case)

$\rightarrow$**:** Deduction Goal $LHS \rightarrow RHS$. Assume LHS. We prove RHS. (Modus Ponens, Inference)

$\exists$**M:** We must construct TM M'. (Proof by Construction)

$\exists$**J:** Let J' denote that stated to exist in LHS. (Give it a name, Existential Instantiation, $\exists$ Elimination)

$\forall$**J:** Because axiom is true for all J, it is true for J'. (Plugging In, Universal Instantiation, $\forall$ Elimination)

$\exists$**M:** Let M' denote that stated to exist in the axiom. (Give it a name, Existential Instantiation, $\exists$ Elimination)

$\forall$**I:** Let I' be arbitrary. (Arbitrary/Worst Case, Universal Generalization, $\forall$ Introduction)

$\forall$**I:** Because axiom and LHS are true for all I, it is true for I'. (Plugging In, Universal Instantiation, $\forall$ Elimination)

=**:** Axiom gives $M'(I')=J'(I')$ and the RHS gives $J'(I')=P'(I')$, and hence by transitivity $M'(I')=P'(I')$.

Proof by Construction: Hence LHS. (Existential Generalization, $\exists$ Introduction)

Deduction: Deduction conclusion $LHS \rightarrow RHS$.

Universal Generalization: Because it was done for a worst case P, it has been done for all P. (Universal Generalization, $\forall$ Introduction)

Logical Consequence: Because it was proved with help from the axiom, it is true in every model/universe in which this axiom is true.

- **Removing Details:** Merging three courses into two needs to remove something. The proposal is to teach them to understand these concepts without going into the details of how one compiles Java programs into TMs.

Logic Game Rules

Operation	Using Oracle Assured Statement	Prover Proving Statement
Exists $\exists$	*Give it a name*	*Proof by Construction*
Formal	**Existential Instantiation** ($\exists$ Elimination)	**Existential Generalization** ($\exists$ Introduction)
Practice	Let y' denote the object assumed to exist in the axiom.	Let y' be 5.
Game	The oracle assures $\exists y\, \alpha(y)$ by providing such a y'.	The prover proves $\exists y\, \alpha(y)$ by constructing such a y'.
Forall $\forall$	*Plugging In*	*Arbitrary/Worst Case*
Formal	**Universal Instantiation** ($\forall$ Elimination)	**Universal Generalization** ($\forall$ Introduction)
Practice	Knowing it is true for all x, it must be true for x'.	Let x' be arbitrary (Adversarially chosen).
Game	The oracle assures $\forall x\, \alpha(x)$ by having the prover give x'.	The prover proves $\forall x\, \alpha(x)$ by having the adversary give x'.
Implies $\rightarrow$	**Modus Ponens / Inference** The oracle assures $\alpha \rightarrow \beta$ by having the prover prove α and then assuring β.	**Deduction** The prover proves $\alpha \rightarrow \beta$ by having the oracle assure him of α and going on to prove β.
Tautology (True in every model)		**Proof** Winning strategy for prover
Negation $\neg$		Reverse roles $\forall$ and $\exists$ hence of prover and adversary