Логические исследования
2019. Т. 25. № 1. С. 100–119
УДК

Logical Investigations
2019, Vol. 25, No. 1, pp. 100–119
DOI: 10.21146/2074-1472-2019-25-1-100-119

Giorgi Japaridze

Computability logic: Giving Caesar what belongs to Caesar

Giorgi Japaridze
Villanova University and Institute of Philosophy Russian Academy of Sciences,
800 Lancaster Avenue, Villanova, PA 19085, USA .
E-mail: giorgi.japaridze@villanova.edu

Abstract: The present article is a brief informal survey of *computability logic* (CoL). This relatively young and still evolving nonclassical logic can be characterized as a formal theory of computability in the same sense as classical logic is a formal theory of truth. In a broader sense, being conceived semantically rather than proof-theoretically, CoL is not just a particular theory but an ambitious and challenging long-term project for redeveloping logic.

In CoL, logical operators stand for operations on computational problems, formulas represent such problems, and their "truth" is seen as algorithmic solvability. In turn, computational problems — understood in their most general, interactive sense — are defined as games played by a machine against its environment, with "algorithmic solvability" meaning existence of a machine which wins the game against any possible behavior of the environment. With this semantics, CoL provides a systematic answer to the question "What can be computed?", just like classical logic is a systematic tool for telling what is true. Furthermore, as it happens, in positive cases "What can be computed" always allows itself to be replaced by "How can be computed", which makes CoL a problem-solving tool.

CoL is a conservative extension of classical first order logic but is otherwise much more expressive than the latter, opening a wide range of new application areas. It relates to intuitionistic and linear logics in a similar fashion, which allows us to say that CoL reconciles and unifies the three traditions of logical thought (and beyond) on the basis of its natural and "universal" game semantics.

Keywords: Computability logic; game semantics; constructive logic; intuitionistic logic; linear logic; interactive computability

For citation: Japaridze G. "Computability logic: Giving Caesar what belongs to Caesar", *Logicheskie Issledovaniya / Logical Investigations*, 2019, Vol. 25, No. 1, pp. 100–119. DOI: 10.21146/2074-1472-2019-25-1-100-119

The present article is essentially a transcript of a lecture I gave in Moscow at the Institute of Philosophy on October 19, 2017. This explains some peculiarities of its style, such as speaking in first person or absence of formal definitions.

Thanks go to my good old friend Vladimir Shalack for organizing the lecture and forcefully nudging me afterwards to write an article based on it.

1. Computability logic versus classical logic

Not to be confused with the generic term "computational logic", "computability logic" (CoL) is the proper name of an approach and ongoing ambitious project initiated by myself back in 2003 [Japaridze, 2003]. I characterize it as a "formal theory of computability in the same sense as classical logic is a formal theory of truth". To see what this means, let us compare the two logics.

- In classical logic, the central semantical concept is *truth*; formulas represent *statements*; and the main utility of the logic is that it provides a systematic answer to the questions "What is (always) *true*?" or "Does *truth* of P (always) follow from *truth* of Q?".[1]

- In computability logic, the central semantical concept is *computability*; formulas represent *computational problems*; and the main utility of the logic is that it provides a systematic answer to the questions "What is (always) *computable*?" or "Does *computability* of P (always) follow from *computability* of Q?".

As we see, the second bulleted item is identical to the first one, only with "truth" replaced by "computability" everywhere and, correspondingly, "statements" by "computational problems" (for computability is the desired property of computational problems just like truth is the desired property of statements). In positive cases computability logic additionally provides a systematic answer to not only questions in the style "*what...*", but also "*how...*", such as "*How* to (always) compute P?", or "*How* to (always) obtain an algorithm for P from an algorithm for Q?". With potential applications in mind, such questions are of course more interesting than their "*what*" style counterparts.

Things are naturally set up so that statements of classical logic turn out to be special cases of computational problems, and classical truth a special case of computability. Eventually this makes classical logic a conservative fragment of CoL: the language of CoL is a proper extension of that of classical logic, but if we limit the former to the latter, CoL validates nothing more and nothing less than what classical logic does.

2. Computability logic versus intuitionistic and linear logics

Similarly, intuitionistic and linear logics can also be viewed as fragments of CoL, albeit "not quite" conservative ones, as CoL validates certain principles

[1] Of course, one is a special case of the other.

not provable in those logics, even if there are more similarities than differences. To me this fact indicates that those two logics are incomplete and do not fully correspond to their underlying philosophies and intuitions.

Let me take the liberty to philosophize a little bit here. I believe the right way to build a new logic is to:

(I) Start with the philosophy and intuitions that we want to capture — call this *informal semantics*.

(II) Then elaborate a *formal semantics* that adequately corresponds to the informal semantics.

(III) And only after that ask what should be provable and what not in a *proof system* for the resulting logic, construct such a system and verify its soundness and completeness.

This is the way classical logic evolved, culminating in Gödel's completeness theorem for first order logic. CoL, too, follows the same pattern. On the other hand, I would say that intuitionistic and linear logics jumped from informal semantics directly into proof systems, skipping the formal semantics phase.

Take Heyting's intuitionistic logic for instance. Its construction started by looking at proof systems for classical logic and removing the postulates that appeared to be wrong from the informal intuitionistic point of view, such as the law of excluded middle.

Similarly, linear logic was obtained from Gentzen's sequent calculus for classical logic as a result of deleting certain rules obviously incompatible with the resource philosophy of linear logic, such as contraction.

Yes, in both cases the underlying philosophical and intuitive considerations were sufficient to clearly see that the expelled principles were indeed wrong. But where is the guarantee that, together with the law of excluded middle or contraction, some innocent, deeply hidden principles did not vanish as well? Idiomatically speaking, where is the guarantee that such a revision of classical logic did not throw out the baby with the bathwater? And, indeed, I dare to argue that this is exactly what happened. In the case of intuitionistic logic, among such "babies" is

$$\begin{array}{c}(\neg P \to A \lor B) \land (\neg Q \to C \lor D) \land \neg(P \land Q) \to \\ (\neg P \to A) \lor (\neg P \to B) \lor (\neg Q \to C) \lor (\neg Q \to D).\end{array} \tag{1}$$

And an example of an innocent victim of rudely rewriting classical logic into linear logic is

$$(A \land B) \lor (C \land D) \to (A \lor C) \land (B \lor D), \tag{2}$$

with its connectives understood in the multiplicative sense. I call the latter Blass's principle as Andreas Blass [Blass, 1992] was the first to study it as an example of a game-semantically vaild principle underivable in linear logic.

Of course some, mostly retroactive, attempts have been made to create formal semantics matching the proof systems of intuitionistic or linear logics. But the reasonable way to go is to match a proof system with a convincing formal semantics rather than vice versa. It is always possible to come up with some formal semantics that matches the target proof system, but the whole question is how adequately and convincingly that semantics captures the philosophy and intuitions underlying the logic.

When constructing a deductive system, we ask what should be provable in it and what not. An answer to this question stems from the underlying semantics and only semantics, formal or informal: those things should be provable that are semantically valid. Some popular approaches to intuitionistic logic have attempted to explain everything in terms of proofs. For instance, you can see the meaning of $A \vee B$ explained by saying that this formula should be considered "good" (true? provable?) if either A or B can be proven. But the whole point is that we are just trying to understand what should be provable and what not. Trying to justify provability in terms of provability creates a vicious circle.

Why is taking a shortcut from the earlier described stage (I) directly to stage (III) wrong? Because it is hardly possible to convincingly argue directly that a given proof system corresponds to a given informal semantics. On the other hand, adequacy (soundness and completeness) of a proof system with respect to a formal semantics can be proven mathematically, as both, unlike informal semantics, are mathematical objects. Now you can ask here: "OK, but where is then the guarantee that the formal semantics adequately captures the informal semantics and thus the original motivations and philosophy underlying the logic?". Of course, there is no guarantee, as this cannot be proven mathematically. But it is easier to argue that the two match each other (when they really do) because both are semantics. Comparing apples with apples is easier than comparing them with oranges.

I have been pushing forward the above points since long ago. While having heard the angry "How dare you!" many times from sympathizers of intuitionistic or linear logics, I am still waiting to see some more convincing attempts to refute them.

Summarizing much of what has been said in this section, my favorite excerpt from [Japaridze, 2009], not without sarcasm, notes:

> *The reason for the failure of the principle of excluded middle in CoL is not that this principle ... is not included in its axioms. Rather, the failure of this principle is exactly the reason why it, or anything*

entailing it, would not be among the axioms of a sound system for CoL.

3. Computational problems as games

Anyway, what is computability? Before trying to answer or even ask this question, one should first understand what a computational problem is, for computability is a property of computational problems. So, what is a *computational problem*? According to Church, a computational problem is nothing but a *function* (to be computed). That is, the task of systematically generating the values of that function at different arguments. The tradition of seeing computational problems as functions has since firmly established in theoretical computer science. Such an approach, however, as acknowledged by Turing [Turing, 1936] himself, is too narrow. Most tasks performed by computers are *interactive*, far from being as simple as just receiving an input and generating an output. For instance, take a look at the work of a network server. It is in fact an infinite process, with signals moving back and forth between it and its environment in a not quite synchronized or regulated fashion, affecting not only current events but some future events as well. Such tasks are not always reducible to functions, at least reducible in some "nice" way. We need something more here, a more general concept to be able to adequately model complex tasks performed by computers.

Such "something more" for us are *games*: a computational problem is a game between a machine, denoted $\top$, and its environment, denoted $\bot$. Then *computability* is understood as existence of a machine which always wins the game, i.e., wins it no matter how the environment acts. In this presentation I am not giving you any formal definitions, including definitions of our concepts of games or game-playing. But such definitions, of course, do exist.

Even though often it is us who act in the role of $\bot$, we are fans of $\top$ rather than $\bot$. That is because $\top$ (machine) is a tool, and its losing the game would mean failing to perform the task it was supposed to perform for us. The behavior (game-playing strategy) of $\top$, as the word "machine" suggests, should be algorithmic as it is a mechanical device. On the other hand, there are no restrictions on the behavior of $\bot$, as the latter represents a capricious user, the blind forces of nature or the devil himself (and you can't ask the devil to only follow algorithmic strategies).

Games can be visualized as trees in the style of Figure 1. Vertices of such a tree represent positions in the game, and edges — their labels, that is — represent legal moves, prefixed with $\top$ or $\bot$ to indicates which player can make the move. On the other hand, the label $\top$ or $\bot$ of a vertex indicates which player is considered to be the winner if the game ends in the corresponding

position. The game can end anywhere, it does not have to continue to the "end": after all, some branches can be infinite and thus there will be nothing that could be understood as the "end". So, if the machine made the move α in the game of Figure 1, the environment responded with γ and no further moves were made, the machine loses as the corresponding vertex of the tree is $\bot$-labeled.

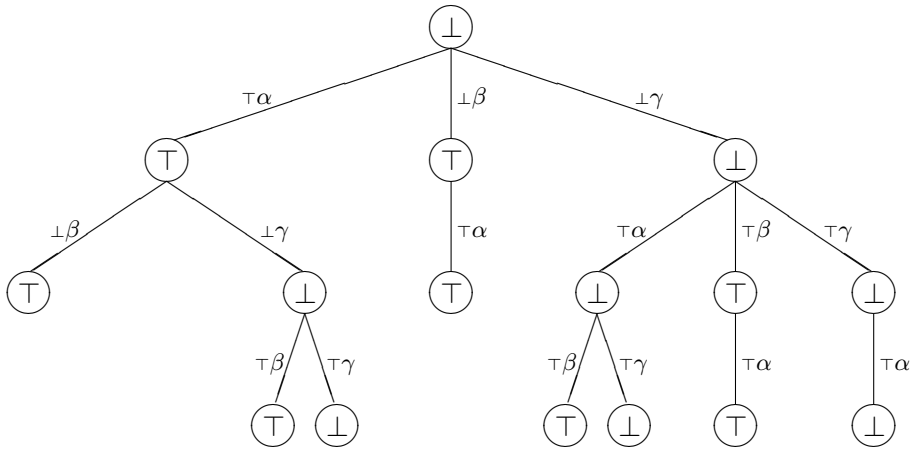

Figure 1: A game of depth 3

Games in logic have been studied by many authors, but our understanding of games is apparently unique in that it does not impose any regulations on the order in which the players should or could move, and permits positions where both players have legal moves. For instance, the root position of the game of Figure 1, as we see, allows either player to move. In that position, a move (if any) will be made by the player which can or want to act faster.

It turns out that, in the sort of games we consider, the relative speed of either player does not matter. Namely, it never hurts a player to postpone making moves and let the adversary go first whenever possible. Such games are said to be *static*, and they are defined by imposing a certain technical yet simple condition on games. Striving to keep this presentation non-technical, I will not discuss that condition here. Suffice it to say that all "pure" (speed-independent) interactive problems turn out to be static, and the class of static games is closed under all game operations studied in CoL. The game of Figure 1 is static, in which the machine has a winning strategy. An interactive algorithm that guarantees the machine a win reads as follows:

> *Regardless of what the adversary is doing or has done, go ahead and make move α; make β as your second (and last) move if and when you see that the adversary has made move γ, no matter whether this happened before or after your first move.*

It is left as an exercise for the reader to see that $\top$, following this interactive algorithm (strategy), wins no matter what and how fast $\bot$ does.

Computational problems in the traditional sense, i.e. functions, are static games of depth 2 of the kind seen in Figure 2.

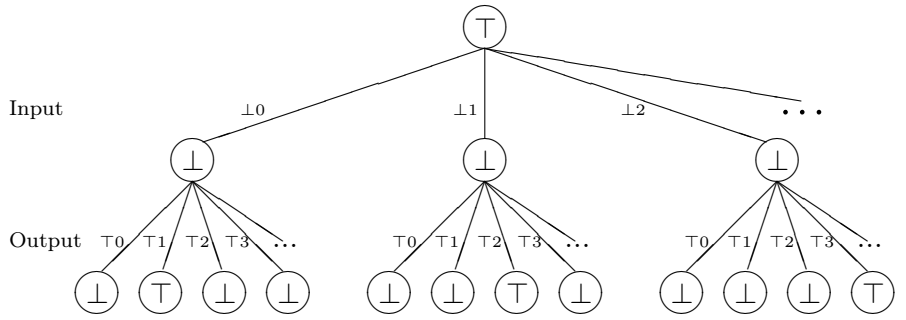

Figure 2: The successor function as a game

In such a game, the upper level edges represent possible inputs provided by the environment, so they are $\bot$-labeled. The lower level edges represent possible outputs generated by the machine, so they are $\top$-labeled. The root is $\top$-labeled because it corresponds to the situation where nothing happened, namely, no input was provided by the environment. The machine has nothing to answer for in this case, so it wins. The middle level nodes are $\bot$-labeled because they correspond to situations where there was an input but the machine failed to generate an output, so the machine loses. Each group of the bottom level nodes has exactly one $\top$-labeled node, because a function has exactly one (correct) value at each argument. It is not hard to see that the particular game of Figure 2 represents the successor function $x+1$: if the input is 0, the machine, in order to win, should generate the output 1, if the input is 1, the output should be 2, etc.

Now CoL rhetorically asks why limit ourselves only to trees of the kind seen in Figure 2. First of all, we may want to allow branches to be longer than 2, or even infinite to be able to model long or infinite tasks performed by computing machines. And why not allow all sorts of distributions of $\top$ and $\bot$ in nodes or on edges? For instance, consider the task of computing the function $3/x$. It would

be natural to make the node to which the input 0 takes us not $\bot$-labeled, but $\top$-labeled. Because the function is not defined at 0, so the machine cannot be held responsible for failing to generate an output on such an input.

It makes sense to generalize computational problems not only in the direction of increasing their depths, but also decreasing. Games of depth 0 are said to be *elementary*. These are games with no legal moves (the game "tree" is just its root), and thus games where one of the players automatically wins by doing nothing. We understand true atomic sentences of classical logic such as $2 \times 2 = 4$ or $\top$ as the elementary game automatically won by the machine, and false sentences such as $2 \times 2 = 5$ or $\bot$ as the elementary game lost by the machine. Note the two different yet related meanings of the symbols $\top$ and $\bot$ in CoL: depending on the context, such a symbol stands either for the corresponding elementary game, or the player which wins that game.

Thus, classical propositions for us are nothing but elementary games. This generalizes to predicates in the standard way. In classical logic, predicates can be thought of as "propositions that (may) depend on variables". Similarly, we allow "games that (may) depend on variables", with predicates being nothing but elementary sorts of such games. As a result, classical logic becomes a special case of CoL — CoL where only elementary games are allowed.

4. Choice operators

Logical operators in CoL stand for operations on games. There is a whole zoo of them, with (at least) four sorts of conjunction and disjunction as well as universal and existential quantifiers, a bunch of so called recurrence (repetition) operations and corresponding implication-style and negation-style operations, and more. In this short presentation we shall only look at the following subset of the logical operators studied in CoL:

$$[\neg, \wedge, \vee, \rightarrow, \forall, \exists, \sqcap, \sqcup, \bigsqcap, \bigsqcup, \mathbin{!}, ?, \circ\!\!-, \circ\!\!\neg\,.]$$

Using the classical notation for the first six of these is no accident. They are conservative generalizations of their classical counterparts from elementary games to all games. Conservative in the sense that, when applied to elementary games (propositions, predicates) only, their extensional meanings and logical behavior turn out to be exactly classical. This is how classical logic naturally becomes a special (elementary) fragment of CoL.

We start with the *choice connectives* $\sqcap$ (conjunction) and $\sqcap$ (disjunction). The way they combine two games A and B to get the new game $A \sqcap B$ or $A \sqcup B$ is depicted in Figure 3.

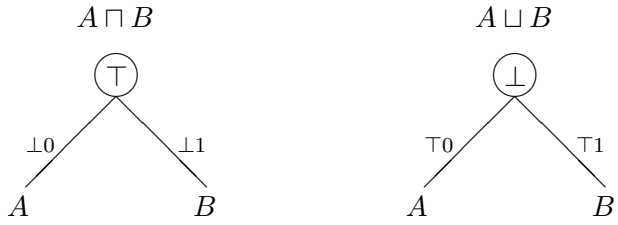

Figure 3: Choice conjunction and disjunction

As we see, $A \sqcap B$ is the game where the first legal move is (only) by the environment. Such a move should be either 0 or 1. If move 0 is made, the game "turns into" A, in the sense that it continues — and the winner is determined — according to the rules of A. Similarly for B in the case of move 1. Intuitively, making move 0 or 1 means choosing between the left disjunct and the right disjunct. Making such a choice is not only a privilege of the environment, but also an obligation: as seen in the picture, the root of $A \sqcap B$ is $\top$-labeled, meaning that the environment loses if it fails to make an initial move/choice.

$A \sqcup B$ is fully symmetric/dual to $A \sqcap B$: in it, it is the machine rather than the environment who makes the initial choice and who loses if no choice is made.

For simplicity, let us agree that the universe of discourse is always $\{0, 1, 2, \cdots\}$. If so, the *choice universal quantification* $\bigsqcap x A(x)$ (note that $\bigsqcap$ is larger than $\sqcap$) can be understood as the infinite choice conjunction $A(0) \sqcap A(1) \sqcap A(2) \sqcap \cdots$, and the *choice existential quantification* $\bigsqcup x A(x)$ as the infinite disjunction $A(0) \sqcup A(1) \sqcup A(2) \sqcup \cdots$. So, now a choice is made not just between 0 or 1, but among $0, 1, 2, \cdots$, as shown in Figure 4.

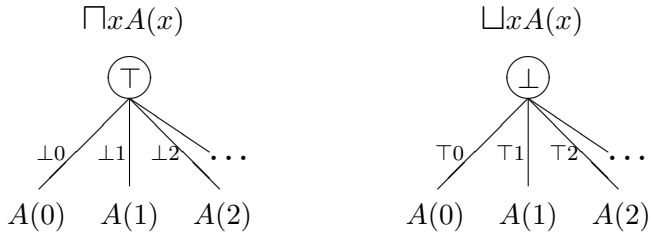

Figure 4: Choice quantifiers

Having these operators in the language, we may now conveniently express standard computational problems (and beyond) without drawing trees. So, for

instance, the problem of computing the successor function depicted in Figure 2 can be simply written as $\sqcap x \sqcup y(y = x + 1)$. In this game, the first move — for instance 2 — is by the environment. Intuitively, this can be seen as asking the machine the question "What is the successor of 2?". The game continues as $\sqcup y(y = 2 + 1)$. The next move — say 3 — is by the machine, which amounts to saying that 3 is the successor of 2. The game is now brought down ("continues as") $3 = 2 + 1$. This is an elementary game with no further moves, and the machine has won because $3 = 2 + 1$ is true. Had the machine made the move 4 instead of 3, or no move at all, it would have lost.

Rather similarly, the problem of deciding a predicate p is expressed by $\sqcap x\big(\neg p(x) \sqcup p(x)\big)$.

5. Negation

Negation $\neg$ is an operation which flips the roles of the two players, turning $\top$'s wins and legal moves into $\bot$'s wins and legal moves, and vice versa. For instance, if *Chess* is the game of chess from the point of view of the white player, then $\neg$*Chess* is the same game as seen by the black player. Figure 5 illustrates how applying $\neg$ to a game A generates the exact "negative image" of A, with $\top$ and $\bot$ interchanged both in the nodes and on the arcs of the game tree.

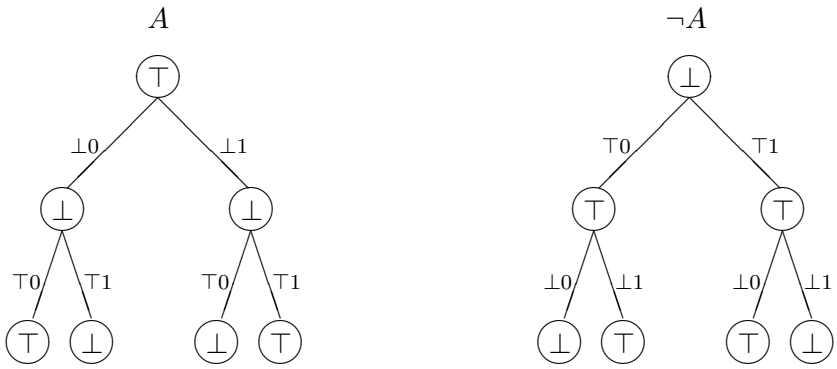

Figure 5: Negation

Obviously if A is a true proposition, i.e., an elementary game automatically won by the machine, then $\neg A$ remains an elementary game but now lost by the machine; in other words, $\neg A$ is a false proposition. This is exactly what was meant when promising that the meaning of $\neg$, or any other operator for which we use classical notation, is exactly classical when limited to elementary games.

It can be easily seen that the games $\neg\neg A$ and A are identical: switching the roles twice brings each player to its original status. Similarly, it can be seen that $\neg$ interacts with choice operations in the kind old DeMorgan fashion. E.g., $\neg(A \sqcap B) = \neg A \sqcup \neg B$. Looking back at Figure 5, notice that the game A shown there is nothing but $(\top \sqcup \bot) \sqcap (\bot \sqcup \top)$, and $\neg A$ is its DeMorgan dual $(\bot \sqcap \top) \sqcup (\top \sqcap \bot)$.

6. Parallel connectives

The operations $\wedge$ and $\vee$ are called *parallel conjunction* and *parallel disjunction*. Unlike their choice counterparts $A \sqcap B$ and $A \sqcup B$, in $A \wedge B$ or $A \vee B$ no choice between A and B is made by either player. Rather, the play proceeds in parallel in both components. To win in $A \wedge B$, the machine should win in both A and B, while for winning $A \vee B$ winning in just one of the two components is sufficient.

Consider, for instance, $\mathit{Chess} \wedge \mathit{Chess}$. This is in fact a play on two boards, where $\top$ plays white on both boards. Perhaps it plays against two adversaries: Peter and Paul, though, for $\top$, they together form just what it calls the (one) environment. In order to win, $\top$ needs to defeat Peter on the left board *and* Paul on the right board. The first move in this compound game is definitely by $\top$, as the opening move is by the white player on both boards. But, after $\top$ makes its first move, say against Peter, the situation changes. Now both $\top$ and its environment naturally have legal moves. Namely, $\top$ has a legal move against Paul, while Peter (and thus the environment from $\top$'s point of view) also has a legal move in response to $\top$'s initial move. It would be unnatural here to impose some regulations regarding which player can go next. This is why CoL's understanding of games does not insist that in each position only either $\top$ or $\bot$ (but not both) should be allowed to move.

To appreciate the difference between the choice and the parallel sorts of connectives, let us compare the two games $\neg\mathit{Chess} \sqcup \mathit{Chess}$ and $\neg\mathit{Chess} \vee \mathit{Chess}$. We assume that draw outcomes are ruled out in *Chess*, and the player who fails to make a move on his turn is considered to have lost. Imagine I am playing in the role of $\top$, and the world champion Kasparov in the role of $\bot$. In $\neg\mathit{Chess} \sqcup \mathit{Chess}$, I have a choice between playing on the left board ($\neg\mathit{Chess}$) or on the right board (Chess). That is, I get to decide whether I want to play black or play white. After such a choice is made, I have to defeat Kasparov on the chosen board, while the other board is discarded. Obviously I stand no chance to win, regardless of whether I choose to play black or white. On the other hand, I can easily beat Kasparov in $\neg\mathit{Chess} \vee \mathit{Chess}$. This is a parallel play on two boards. At the beginning, both Kasparov and I have legal moves: Kasparov on the left board where he is playing white, and I on the right board.

Rather than hurrying to make an opening move, I wait to let Kasparov move first. If he, too, chooses to do nothing, then I win due to being the winner on the left board. Now suppose Kasparov makes his opening move on the left board. Can you guess how I should respond? Yes, by making the exact same move on the right board. I wait again. Whatever move Kasparov makes on the right board in response, I copy that move back on the left board. And so on. By using this copy-cat strategy, I am in fact letting Kasparov play against himself. Eventually, both he and I are guaranteed to win on one board and lose on the other. Since this is a disjunction, having won in one of the disjuncts makes me the winner in the overall game.

In general, the law of excluded middle "$\neg A$ OR A" is invalid in CoL with OR understood as $\sqcup$ but valid when OR is understood as $\vee$: one can prove that, while the above seen copy-cat strategy wins all games of the form $\neg A \vee A$, for some A no machine can win $\neg A \sqcup A$ against a sufficiently smart adversary.

7. Putting things where they belong

What is meant by "Giving Caesar what belongs to Caesar" (... and God what belongs to God) in the title of this article? The twentieth century has witnessed endless and fruitless fights between the classically-minded and the constructivistically-minded regarding whether the law of excluded middle should be accepted or rejected. It is obvious that the two schools of thought were talking about two very different meanings of disjunction. Yet, for some strange reason, they chose the same symbol $\vee$ for both, and then started arguing with each other. Not quite serious I would say. CoL neutralizes this and similar controversies by putting things where they belong. And, as pointed out in Section 2., it does so *semantically*, not because it allows or forbids them among the postulates of some purportedly "right" deductive system.

Give the classically minded what belongs to the classically-minded ($\vee$), and the constructivists what belongs to the constructivists ($\sqcup$)!

- Yes, classical logic is right: $\neg A \vee A$ is indeed valid.
- Yes, intuitionistic logic is right: $\neg A \sqcup A$ is indeed invalid.

No subject for arguing!

The classical tautology $(\neg A \wedge \neg A) \vee A$ fails in CoL unless A is stipulated to be elementary. Observe that, at least, the copy-cat trick used earlier in our winning strategy for $\neg Chess \vee Chess$ no longer works for the "similar" $(\neg Chess \wedge \neg Chess) \vee Chess$. I can try to copy Kasparov's moves in *Chess* within both conjuncts of $\neg Chess \wedge \neg Chess$ and vice versa. However, Kasparov may start acting in different ways in these two conjuncts, and then, at best, I will be able

to synchronize only one of them with *Chess*. It is then possible that eventually I lose in *Chess* and in the unsynchronized conjunct of $\neg Chess \wedge \neg Chess$, which makes me lose in the overall game $(\neg Chess \wedge \neg Chess) \vee Chess$. Anyway, classical logic accepts the principle $(\neg A \wedge \neg A) \vee A$ and linear logic rejects it (with $\wedge, \vee$ seen as multiplicatives). Which one is "right"?

The formal language of pure CoL has two sorts of nonlogical atoms: *elementary* and *general*. Elementary atoms are meant to be interpreted as elementary games, and general atoms as any games, elementary or not. We use the lowercase $p, q, \cdots$ for elementary atoms and the uppercase $P, Q, \cdots$ for general atoms.

And, again, Caesar is being given what belongs to Caesar and God what belongs to God. The semantics of CoL classifies:

- $(\neg p \wedge \neg p) \vee p$ as valid. Yes, classical logic is right!
- $(\neg P \wedge \neg P) \vee P$ as invalid. Yes, linear logic is right!

(As for the earlier discussed law of excluded middle, both $\neg P \vee P$ and $\neg p \vee p$ are valid and both $\neg P \sqcup P$ and $\neg p \sqcup p$ are invalid.)

From CoL's perspective, classical logic differs from intuitionistic logic in its understanding of logical constants (operators), and differs from linear logic in its understanding of logical variables (nonlogical atoms).

8. Reduction

The *implication* operation $\rightarrow$ is defined in the standard way by

$$A \rightarrow B \ =_{def} \neg A \vee B.$$

The intuition associated with this operation is that of a *reduction* of the consequent to the antecedent. Since A is negated here and thus the roles of the two players are interchanged in it, A can be seen by $\top$ as an environment-provided *resource* rather than a task. Namely, $\top$ can observe how the environment is playing in A and use that information in its play in B. The task of $\top$ is to win B as long as the environment wins A; in other words, to solve problem B as long as the environment is (correctly) solving problem A.

To get a feel of $\rightarrow$ as a reduction operation, consider the game

$$\begin{array}{c} \sqcap x \sqcup y\big(y = Father(x)\big) \wedge \sqcap x \sqcup y\big(y = Mother(x)\big) \\ \rightarrow \ \sqcap x \sqcup y\big(y = PaternalGrandmother(x)\big), \end{array}$$

where $Father(x)$ is the function "x's father", and similarly for $Mother(x)$ and $PaternalGrandmother(x)$. Here, the task $\top$ is facing is telling the name of an

arbitrary person's paternal grandmother while the environment (correctly) tells the name of an arbitrary person's father and the name of an arbitrary person's mother. In other words, this is the problem of reducing the paternal grandmotherhood problem to the fatherhood and motherhood problems. Winning this game is easy and does not require any knowledge of anyone's relative's names. Here is a strategy for $\top$: Wait till $\bot$ makes a move a in the consequent (if not, $\top$ wins automatically). Intuitively, such a move amounts to asking $\top$ the question "Who is a's paternal grandmother?". Make the same move a in the first conjunct of the antecedent, i.e., ask the counterquestion "Who is a's father?". $\bot$ will have to answer correctly, or else it loses. Let us say $\bot$'s answer/move is b. Make the same move b in the second conjunct of the antecedent, thus asking $\bot$ to tell who b's mother is. $\bot$, again, will have to provide the correct answer, let us say c. Now, by making the same move c in the consequent, i.e., answering "c" to $\bot$'s original question regarding a's paternal grandmother, $\top$ wins: c is indeed a's paternal grandmother (unless the environment lied in the antecedent about a's father or b's mother, but in that case, as already noted, $\top$ is no longer responsible for anything).

9. Blind quantifiers

The operations $\forall$ and $\exists$, called *blind quantifiers*, conservatively generalize their classical counterparts, just like $\neg, \wedge, \vee, \rightarrow$ do. Unlike the choice quantifiers, there are no moves associated with $\forall$ or its dual $\exists$. Playing $\forall x A(x)$ or $\exists x A(x)$ means playing $A(x)$ "blindly", without knowing the value of x as the latter is not specified by either player. In order to win $\forall x A(x)$ (resp. $\exists x A(x)$), $\top$ needs to play $A(x)$ in such a way that it wins for all (resp. at least one) possible values of x.

An alternative intuitive characterization of $\forall x A(x)$ and $\exists x A(x)$ would be that, in these games, a third party chooses a value for x but never shows it to either player. In order to win $\forall x A(x)$ (resp. $\exists x A(x)$), $\top$ (resp. $\bot$) needs to play $A(x)$ in a way that guarantees success regardless of what that chosen value might have been.

Let us compare the games

$$\sqcap x\big(\mathit{Even}(x) \sqcup \mathit{Odd}(x)\big) \quad \text{and} \quad \forall x\big(\mathit{Even}(x) \sqcup \mathit{Odd}(x)\big).$$

$\sqcap x\big(\mathit{Even}(x) \sqcup \mathit{Odd}(x)\big)$, which is a game of depth 2, is easy to win: wait till the adversary selects a value m for x; if m is even, respond by choosing the left disjunct of $\mathit{Even}(m) \sqcup \mathit{Odd}(m)$, otherwise respond by choosing the right disjunct, and rest your case. On the other hand, $\forall x\big(\mathit{Even}(x) \sqcup \mathit{Odd}(x)\big)$ is a game of depth 1, and it is impossible to win. Here the value of x is not specified by the adversary or whoever for that matter, yet you should do the impossible

task of choosing between $Even(x)$ and $Odd(x)$ so that all of the elementary games/propositions $Even(0), Even(1), Even(2), \cdots$ (if you chose $Even(x)$) or $Odd(0), Odd(1), Odd(2), \cdots$ (if you chose $Odd(x)$) are won/true.

This should not suggest than all $\forall$-games are unwinnable. Consider

$$\forall x\Big(Even(x) \sqcup Odd(x) \rightarrow \sqcap y\big(Even(x+y) \sqcup Odd(x+y)\big)\Big).$$

Here, given a number chosen by the environment for y, let us say 5, in order to tell whether $x+5$ is even or odd it is not necessary to know the actual value of x. Rather, just knowing whether x is even or odd is sufficient. And, luckily, this piece of information on x will have to be provided by the environment as mandated by $Even(x) \sqcup Odd(x)$ in the antecedent. If the environment claims that x is even, then $\top$ chooses $Odd(x+5)$ and wins; otherwise, it chooses $Even(x+5)$.

$\forall$ can be seen to be stronger than $\sqcap$, in the sense that the semantics of CoL validates the principle $\forall x A(x) \rightarrow \sqcap x A(x)$ but not its contrapositive. This means that $\sqcap x A(x)$ is reducible to $\forall x A(x)$ but not vice versa. Symmetrically, $\exists$ is weaker than $\sqcup$.

Speaking philosophically, choice quantifiers are *constructive* versions of their blind counterparts. While not as popular as the law of excluded middle, $\exists x \forall y\big(p(x) \vee \neg p(y)\big)$ is another example of a valid principle of classical logic which, however, is not valid in any constructive sense, and not provable in intuitionistic logic. Again giving Caesar what belongs to Caesar, CoL unsurprisingly establishes:

- Both $\exists x \forall y\big(p(x) \vee \neg p(y)\big)$ and $\exists x \forall y\big(P(x) \vee \neg P(y)\big)$ are valid. Yes, classical logic is right!
- Both $\sqcup x \sqcap y\big(p(x) \vee \neg p(y)\big)$ and $\sqcup x \sqcap y\big(P(x) \vee \neg P(y)\big)$ are invalid. Yes, intuitionistic logic is right!

On the other hand, the valid principle $\forall y \exists x\big(p(x) \vee \neg p(y)\big)$ of classical logic is commonly recognized to be valid in every reasonable constructive sense, and is provable in intuitionistic logic. As expected, CoL validates this principle with both (blind and choice) sorts of quantifiers and both (elementary and general) sorts of atoms.

10. Recurrences

Out of several types of so called *recurrence* operations studied within the framework of CoL, here we shall only take a look at *branching recurrence* $⍦$. Its dual *corecurrence* operation $⍧$ can simply be understood as $\neg ⍦ \neg$. When applied to a game G, $⍦$ turns it into a game playing which means repeatedly playing G.

When G is seen as a resource (e.g., when it is in the antecedent of an implication), $⍟$ generates multiple "copies" of G, thus making G a reusable/recyclable resource.

In classical logic, this sort of an operation would be meaningless, because classical logic is resource-blind, seeing no difference between one and many copies of G. In the resource-conscious CoL, however, recurrence operations are not only meaningful, but also necessary to achieve a satisfactory level of expressiveness and realize CoL's potential and ambitions. Hardly any computer program is used only once; rather, it is run over and over again. Loops within such programs also assume multiple repetitions of the same subroutine. In general, the tasks performed in real life by computers, robots or humans are typically recurring ones or involve recurring subtasks.

Let me use our old friend *Chess* to explain the meaning of $⍟$. A play of $⍟$*Chess* starts as an ordinary play of *Chess*. At any time, however, the environment may decide to split the current position into two identical ones, thus creating two runs of *Chess* out of one that have a common past but possibly diverging futures. From that point on, the play continues on two boards. At any time, the environment can again create two identical copies of the then-current position on either board, and the play correspondingly continues on three boards. The environment can keep splitting positions in this fashion, creating more and more sessions of *Chess* to be played in parallel. Eventually, $\top$ will be considered the winner if it wins in all of those sessions. $⍰$*Chess* is similar, with the difference that now splitting positions is $\top$'s privilege, and $\top$ wins if it wins in at least one of the multiple sessions of *Chess*.

11. Brimplication

The implication-style operation $\circ\!\!-$, called *brimplication* ("br" for "branching"), is defined by

$$A \circ\!\!- B \ =_{def} ⍟A \rightarrow B.$$

Intuitively $A \circ\!\!- B$, just like $A \rightarrow B$, is a problem of reducing B to A. The difference between the two reduction operations is that, while in $A \rightarrow B$ the machine has a single copy of A available as an environment-provided informational resource for solving B, in $A \circ\!\!- B$ the resource A — as well as any game/position it has evolved to — can be duplicated and reused any number of times. As a result, $A \circ\!\!- B$ is easier for $\top$ to win than $A \rightarrow B$ because, as a resource, the antecedent of $A \circ\!\!- B$ is stronger (very much so) than the antecedent of $A \rightarrow B$. While being the most *basic* sort of reduction allowing us to naturally define $\circ\!\!-$ or other reduction-style operations, $\rightarrow$ is a stricter and thus less general operation of reduction than $\circ\!\!-$. In fact, according to Thesis 1 below, $\circ\!\!-$ is *the most general* sort of reduction.

We say that a problem/game B is *brimplicatively reducible* to a problem A iff there is a machine with a winning strategy for $A \circ\!\!- B$.

Thesis 1. *Brimplicative reducibility is an adequate mathematical counterpart of our intuition of reducibility in the weakest — and hence the most general — algorithmic sense possible. Namely, for all games/problems A and B, we have:*

(I): *Whenever B is brimplicatively reducible A, B is also algorithmically reducible to A according to everyone's reasonable intuition.*

(II): *Whenever B is algorithmically reducible to A according to everyone's reasonable intuition, B is also brimplicatively reducible to A.*

This is pretty much in the same sense as, by Church's celebrated thesis, a function f is Turing-machine computable iff f is algorithmically computable according to everyone's reasonable intuition.

It should be also mentioned that, unsurprisingly, brimplicative reducibility turns out to be a conservative generalization of Turing reducibility, commonly accepted in theoretical computer science as the most general relation of reducibility between the traditional, non-interactive sorts of problems.

12. On intuitionistic logic once again

According to Kolmogorov's [Kolmogorov, 1932] well known thesis, intuitionistic logic is a logic of problems. This thesis was stated by Kolmogorov in rather abstract, philosophical terms. No past attempts to find a strict and adequate mathematical explication of it have fully succeeded. The following theorem tells a partial success story ("partial" because it is limited to only positive propositional fragment of intuitionistic logic):

Theorem 1. *[Japaridze [Japaridze, 2007b]; Mezhirov and Vereshchagin [Mezhirov, Vereshchagin, 2010]] The positive (negation-free) propositional fragment of Heyting's intuitionistic calculus is sound and complete with respect to the semantics of CoL, with intuitionistic implication understood as $\circ\!\!-$, conjunction as $\sqcap$ and disjunction as $\sqcup$.*

As for the intuitionistic operators not mentioned in the above theorem, CoL sees the intuitionistic universal quantifier as $\bigsqcap$, existential quantifier as $\bigsqcup$, and negation as what it calls *brefutation* $\circ\!\!\neg$, defined by

$$\circ\!\!\neg A \ =_{def} \ A \circ\!\!- \bot.^2$$

[2] As we remember from Section 3., the meaning of the logical constant $\bot$ in CoL is standard: this is an always-false proposition, i.e., the elementary game automatically lost by the machine.

So, formula (1) from Section 2. should in fact have been written as

$$\begin{array}{c}(\circ\!\!\neg P \circ\!\!- A \sqcup B) \sqcap (\circ\!\!\neg Q \circ\!\!- C \sqcup D) \sqcap \circ\!\!\neg (P \sqcap Q) \circ\!\!- \\ (\circ\!\!\neg P \circ\!\!- A) \sqcup (\circ\!\!\neg P \circ\!\!- B) \sqcup (\circ\!\!\neg Q \circ\!\!- C) \sqcup (\circ\!\!\neg Q \circ\!\!- D).\end{array} \tag{3}$$

This formula, as noted earlier, is valid in CoL but unprovable in Heyting's calculus, making the latter incomplete with respect to the semantics of CoL. At the same time, Heyting's calculus in its full first order language has been shown [Japaridze, 2007a] to be sound with respect to CoL's semantics. So, intuitionistic logic — at least, Heyting's formal version of it — is a fragment of CoL but, unlike classical logic, "not quite" a conservative one. Nevertheless, since (3) is the shortest formula known to separate Heyting's calculus from the corresponding fragment of CoL, one can say that Heyting's calculus is quite close to being complete.

13. Conclusion

Computability logic (CoL) is a formal theory of computability in the same sense as classical logic is a formal theory of truth. Its formulas represent computational problems, logical operators stand for operations on such problems, and validity means being "always computable". Computational problems, in turn, are understood in their most general — interactive — sense and, mathematically, are defined as games played by a machine against its environment.

This article was a brief, informal and incomplete survey of CoL. The latter is not a subject that can be duly introduced within a 1-hour presentation and, in order to well understand it, one will have to use additional sources. Out of the numerous publications devoted to CoL, the most recommended reading for a beginner are the first ten sections of [Japaridze, 2009]. An even more comprehensive — and the most up-to-date — survey of CoL can be found online in [Japaridze, 20019].

There was no discussion of related literature in this article. Such discussions can be found elsewhere, including the already mentioned [Japaridze, 2009] or [Japaridze, 20019]. I just want to point out here that the main precursors of CoL are Lorenzen's [Lorenzen, 1961] dialogue semantics for intuitionistic logic, Hintikka's [Hintikka, 1973] game-theoretic semantics for classical logic and Blass's [Blass, 1992] game semantics for linear logic, the latter being the closest one.

The language of CoL is much more expressive than the fragment surveyed in the present article. Important topics not covered here also include the proof theory of CoL. And, of course, actual and potential applications of CoL outside logic itself. Such applications include theory of (interactive) computation,

knowledgebase systems, systems for planning and action, declarative programming languages, constructive applied theories, and more.

So far the most manifestly realized extra-logical utility of CoL has been using it as a logical basis for applied theories [Japaridze, 20010]-[Japaridze, 20016c], with such theories offering substantial advantages over their classical-logic-based counterparts. CoL-based number theory, termed *clarithmetic*, will be the subject of a forthcoming paper expected to appear in the next issue of this journal.

I want to close this article by pointing out that, despite having been evolving for 15 years already, CoL, due to its ambitiousness, still remains at an early stage of development, with more open questions than answered ones. A researcher who decides to join the project will find enough interesting material to be occupied with for many years to come. Students are especially encouraged to try.

References

Blass, 1992 – Blass, A. "A game semantics for linear logic", *Annals of Pure and Applied Logic*, 1992, Vol. 56, pp. 183–220.

Hintikka, 1973 – Hintikka, J. *Logic, Language-Games and Information: Kantian Themes in the Philosophy of Logic*. Clarendon Press, 1973.

Japaridze, 2003 – Japaridze, G. "Introduction to computability logic", *Annals of Pure and Applied Logic*, 2003, Vol. 123, pp. 1–99.

Japaridze, 2007a – Japaridze, G. "Introduction to computability logic", *Acta Cybernetica*, 2007, Vol. 18, No. 1, pp. 77–113.

Japaridze, 2007b – Japaridze, G. "The intuitionistic fragment of computability logic at the propositional level", *Annals of Pure and Applied Logic*, 2007, Vol. 143, No. 1, pp. 187–227.

Japaridze, 2009 – Japaridze, G. "In the beginning was game semantics", in: *Games: Unifying Logic, Language, and Philosophy*, eds. by O. Majer, A.-V. Pietarinen and T. Tulenheimo. Springer, 2009, pp. 249–350.

Japaridze, 20010 – Japaridze, G. "Towards applied theories based on computability logic", *Journal of Symbolic Logic*, 2010, Vol. 75, pp. 565–601.

Japaridze, 20011 – Japaridze, G. "Introduction to clarithmetic I", *nformation and Computation*, 2011, Vol. 209, pp. 1312–1354.

Japaridze, 20014 – Japaridze, G. "Introduction to clarithmetic III", *Annals of Pure and Applied Logic*, 2014, Vol. 165, pp. 241–252.

Japaridze, 20016a – Japaridze, G. "Introduction to clarithmetic II", *Information and Computation*, 2016, Vol. 247, pp. 290–312.

Japaridze, 20016b – Japaridze, G. "Build your own clarithmetic I: Setup and completeness", *Logical Methods in Computer Science*, 2016, Vol. 12, Issue 3, Paper 8, pp. 1–59.

Japaridze, 20016c – Japaridze, G. "Build your own clarithmetic II: Soundness", *Logical Methods in Computer Science*, 2016, Vol. 12, Issue 3, Paper 12, pp. 1–62.

Japaridze, 20019 – Japaridze, G. "Computability Logic Homepage", *An Online Survey of Computability Logic*. 2019. [`www.csc.villanova.edu/$\sim$japaridz/CL/`, accessed on 21.02.2019]

Kolmogorov, 1932 – Kolmogorov, A. N. "Zur Deutung der intuitionistischen Logik", *Mathematische Zeitschrift*, 1932, Vol. 35, pp. 58–65.

Lorenzen, 1961 – Lorenzen, P. "Ein dialogisches Konstruktivitätskriterium", in: *Infinitistic Methods*. PWN, Proc. Symp. Foundations of Mathematics., Warsaw, 1961, pp. 193–200.

Mezhirov, Vereshchagin, 2010 – Mezhirov, I., Vereshchagin, N. "On abstract resource semantics and computability logic", *Journal of Computer and Systems Sciences*, 2010, Vol. 76, pp. 356–372.

Turing, 1936 – Turing, A. "On Computable numbers with an application to the entsheidungsproblem", *Proceedings of the London Mathematical Society*, 1936, Vol. 2:42, pp. 230–265.