Between two- and three-variable logic over word models

Kamal Lodaya
with Andreas Krebs, Paritosh Pandya, Howard Straubing

The Institute of Mathematical Sciences, Chennai

July 2017

First-order and temporal logic

- First-order logic *FO* is normally interpreted over structures with domains D
- Linear-time temporal logic is normally interpreted over linearly ordered structures $(D, <)$
- Linear temporal logic *LTL* is normally interpreted over $(\omega, <, Suc)$, called ω-words over A, a finite alphabet
  $$\alpha ::= \quad p \in Prop \mid \neg\, \alpha \mid \alpha \ \vee \ \beta \mid \mathsf{X}\alpha \mid \mathsf{F}\alpha \mid \mathsf{Y}\alpha \mid \mathsf{P}\alpha \mid \alpha\ \mathsf{U}\ \beta \mid \alpha\ \mathsf{S}\ \beta$$
- Think of the letters of the alphabet as interpretations of atomic propositions (monadic predicates), A is $\wp(Prop)$
- $w, i \models p$ iff $p \in w(i)$
- $w, i \models \mathsf{X}\alpha$ iff $w, i+1 \models \alpha$
- $w, i \models \alpha \mathsf{U} \beta$ iff for some $k : i \leq k : w, k \models \beta$ and $\forall j : i \leq j < k : w, j \models \alpha$

Finite models

- Here we consider *FO* and *LTL* (with enhancements) interpreted over finite models which are initial segments of ω, that is, $(\{1, \ldots, n\}, <, Suc)$, called finite words over a finite alphabet A, or atomic propositions $\wp(Prop)$
- Sets of finite words are called (formal) languages
- *FO* sentences and *LTL* formulae define languages
- Use regular expression notation, for example $(ab)^* aa(ab)^*$ over the alphabet {a,b} is words with alternating occurrences of the two letters beginning with a and ending with b, except for one consecutive occurrence of a's somewhere in the middle
- Results extend to ω-words

Quantifier alternation hierarchy (Brzozowski-Cohen, Thomas)

- An *FO* sentence is $\Sigma_r[<]/\Pi_r[<]$ if it has r alternating blocks of quantifiers, with first block existential/universal
- $\Delta_r[<]$ is the class of languages which are definable by both $\Sigma_r[<]$ and $\Pi_r[<]$ sentences ($\Sigma_r[<] \cap \Pi_r[<]$ languages)
- Language A^*aaA^* in $\Sigma_2[<] \setminus \Pi_2[<]$, defined by $\exists x \exists y \forall z (x < z < y \supset \bigwedge_{a \in A} \neg a(z))$
- $(ab)^*$ in $\Pi_2[<] \setminus \Sigma_2[<]$
- $(a(ba)^*b(ba)^*)^*$ in $\Pi_3[<] \setminus \mathcal{B}_2[<]$

Finite-variable logics

- FO^k, FO using at most k variables
- (Kamp) introduced binary modalities such as “until” to capture FO over linear orders
- By a standard translation reusing the same bound variables, this means that $FO[<]$ is equivalent to FO^3 (Kamp)

Two-variable logic

- (Mortimer, Grädel-Otto-Rosen) showed that FO^2 is much weaker than FO
- (Thérien-Wilke) showed that over words, $FO^2[<]$ is below the second level of the quantifier alternation hierarchy
- $FO^2[<]$ is exactly $\Delta_2[<] = \Sigma_2[<] \cap \Pi_2[<]$ (Thérien-Wilke)
- With alphabet $\{0,1\}^3$, let ADD be the language of words representing vertically three numbers $\begin{pmatrix} m \\ n \\ m+n \end{pmatrix}$
- ADD is not definable in $FO^2[<]$

During and throughout (Pratt, Segerberg)

- During the flight from Delhi to Seoul they serve two snacks: $\#snack(del, seoul) = 2$
- I slept throughout the flight from Chennai to Delhi: $\neg awake(ch, del)$, or $\#awake(ch, del) = 0$
- Generally speaking, counting propositions upto a threshold
- These operators also appeared in interval temporal logic and duration calculus (Moszkowski, Zhou-Hoare-Ravn)

Within first-order logic

- Thresholds are easily defined in first-order logic on words:
  $\#snack(del, seoul) = 2$ translates to
  $del < seoul \wedge$
  $\exists x, y : del < x < y < seoul \wedge snack(x) \wedge snack(y) \wedge$
  $\forall z : (del < z < x) \vee (x < z < y) \vee (y < z < seoul) \supset$
  $\neg snack(z)$
- Oriented: $\#snack(seoul, del) = 2$ is about the return flight
- So $x < y$ is definable as $\#A(x, y) \geq 0$ (A is the alphabet)
- Translating out thresholds requires more variables
- Thresholds are not definable in two-variable logic FO^2

Counting upto a threshold

- A threshold constraint has the form: $\#B(x, y) \sim c$, where $B \subseteq A$, $\sim$ a comparison operation and $c \geq 0$
- Its semantics is as follows: $w, i, j \models \#B(x, y) \sim c \iff |\{z \mid x < z < y \wedge \bigvee_{a \in B} a(z)\}| \sim c$

Logics with threshold

- The logic $FO[Th]$ with threshold constraints has the same expressiveness as $FO[<]$
- The logic $FO^2[Th]$ is the two-variable fragment
- The temporal logic $ThTL[\mathsf{F}, \mathsf{P}]$ has unary future and past modalities $\mathsf{F}_{\#B\sim c}$ and $\mathsf{P}_{\#B\sim c}$
- $w, i \models a$ iff $w(i) = a$
- $w, i \models \mathsf{F}_{\#B\sim c}\alpha$ iff for some $k > i$, we have: $w, k \models \alpha$ and $|\{j \mid w(j) \in B,\ i < j < k\}| \sim c$

Two-variable logic with between

- The binary between relation $a(x, y)$ is defined as the threshold constraint $\#\{a\}(x, y) > 0$
  (we usually drop the set brackets)
- $w, i, j \models a(x, y)$ iff for some z, $x < z < y$ and $w(z) = a$
- The logic $FO[Bet]$ with only between constraints has the same expressiveness as $FO[<]$
- The logic $FO^2[Bet]$ is the two-variable fragment only using between constraints

Theorem
$FO^2[Th]$ and $FO^2[Bet]$ are equally expressive, their sentences define the same class of languages.

Definability examples

- The successor relation $Suc(x, y)$ is definable as $\#A(x, y) = 0$
- Hence languages like A^*aaA^* which are not $FO^2[<]$-definable can be expressed
- But one can go beyond $FO^2[<, Suc]$: $Stair_k = A^*ac^*ac^* \dots c^*aA^*$ with k intermediate occurrences of a (Etessami-Wilke) can be expressed, these are at different levels of the until-since hierarchy of linear temporal logic (Thérien-Wilke)
- Includes $\mathcal{B}_2[<]$, the second level of the quantifier alternation hierarchy

Implementing numbers

- An r-bit value $b_1 \cdots b_r$ over $\{0,1\}$, from LSB b_1 to MSB b_r:

$$\begin{aligned} Bit_0^1(x) &\stackrel{\text{def}}{=} \exists y.\ Suc(x,y) \wedge 0(y) \\ Bit_0^{i+1}(x) &\stackrel{\text{def}}{=} \exists y.\ Suc(x,y) \wedge Bit_0^i(y) \end{aligned}$$

(similarly $Bit_1^i(x)$)

- A sequence $c_1 c_2 \ldots c_k$ of r-bit values with separator s:
  $s \cdot b_1^1 \cdots b_r^1 \cdot s \cdot b_1^2 \cdots b_r^2 \cdot s \cdots s \cdot b_1^k \cdots b_r^k$
- Value at x and value at y are the same (language is $A^* s b_1 \ldots b_r A^* s b_1 \ldots b_r A^*$):

$$\begin{aligned} EQ_i(x,y) &\stackrel{\text{def}}{=} (Bit_0^i(x) \Leftrightarrow Bit_0^i(y)) \\ EQ(x,y) &\stackrel{\text{def}}{=} s(x) \wedge s(y) \wedge \bigwedge_{i=1}^{r} EQ_i(x,y) \end{aligned}$$

Operations on numbers

- (Value at y) is (value at x)+1: language is
$\bigcup_{i=1}^{r} A^* s 1^{i-1} 0 b_{i+1} \dots b_r A^* s 0^{i-1} 1 b_{i+1} \dots b_r A^*$

$$INC1(x,y) \stackrel{\text{def}}{=} \bigvee_i \quad (Bit_0^i(x) \wedge \bigwedge_{h=1}^{i-1} Bit_1^h(x)) \supset$$
$$(Bit_1^i(y) \wedge \bigwedge_{h=1}^{i-1} Bit_0^h(y) \wedge \bigwedge_{j=i+1}^{r} EQ_j(x,y))$$

- (Value at y) is $2\times$(value at x): language is $A^* s 0^h 1 1^i 0 b_{h+i+3} \dots b_r A^* s 0^h 0 1^i 1 b_{h+i+3} \dots b_r A^*$, together with a matching of the border positions, when value of x is below 2^{r-1}

Defining counters

- Consider a threshold constraint $\#a(x, y) = 2^r$
- Expand the alphabet to $A \times \{0, 1\}^r$
  ("vertical" layout rather than "horizontal")
- An r-bit counter:
  $\forall x, y.\ Suc(x, y) \supset (a(x) \supset INC1(x, y)) \wedge (\neg a(x) \supset EQ(x, y))$
- Observe that everything is done in $FO^2[<, Suc]$,
  because we fixed the length of the numbers

Complexity of satisfiability

- Three colour predicates red, blue and green
- The colour at the beginning of the word, and for the first 2^r occurrences of a, is red
- For the next 2^r occurrences of a is blue, then green ...
- Change the colour cyclically red $\rightarrow$ blue $\rightarrow$ green $\rightarrow$ red each time the counter resets to zero by overflowing
- Replace constraint by an equisatisfiable $FO^2[Bet]$ formula: $a(x,y) \wedge EQ(x,y) \wedge (\neg red(x,y) \vee \neg blue(x,y) \vee \neg green(x,y))$

Theorem

Satisfiability of $FO^2[Th]$ polynomially reduces to satisfiability of $FO^2[Bet]$. Satisfiability of $FO^2[Bet]$ is Expspace-complete.

Addition defines betweenness

- Alphabet $\{0,1\}^2$ ("vertical" layout)
- Let $\mathit{Double}(x,y)$ be a "doubling" predicate, specifying that the positions from x to y (both included) represent vertically two numbers $\binom{n}{2\times n}$
- Consider $a(x) \wedge b(y) \wedge \neg(C \setminus \{c\})(x,y)$, defining the word ac^*b from positions x to y
- By mapping the letter a to $\binom{1}{0}$, b to $\binom{0}{1}$ and c to $\binom{1}{1}$, reduce betweenness—rather its complement "throughout"—to $\binom{11^*0}{01^*1}$ with rows of the same length
- That is: some numbers $\binom{n}{2\times n}$ defined by the formula $\binom{1}{0}(x) \wedge \binom{0}{1}(y) \wedge \mathit{Double}(x,y)$

Addition defines betweenness, continued

- From position x to position y,
  language ac^*b is simulated by doubling,
  cc^*b and ac^*c are simulated by increment,
  and cc^*c is simulated by equality
- All these are over alphabet $\{0,1\}^2$
- Hence betweenness is definable using a binary predicate $Add(x,y)$ representing addition (over alphabet $\{0,1\}^3$) from position x to y (both included) of the form $\begin{pmatrix} m \\ n \\ m+n \end{pmatrix}$

Defining addition (Chandra-Fortune-Lipton)

- With alphabet $\{0,1\}^3$, let *ADD* be the language of words representing vertically three numbers $\begin{pmatrix} m \\ n \\ m+n \end{pmatrix}$
- Correctness: *i*'th bit of sum from *i*'th bit of inputs and whether or not there is a carry into bit *i*
- If *i*'th bit of both inputs is 1, we map the letter to a (set); if *i*'th bit of both inputs is 0, to b (reset); else to c (neutral); think of a monoid {a,b,c}
- Carry product $x \cdot a = a$, $x \cdot b = b$, $x \cdot c = x$ (monoid U2)
- Carry into bit $i+1$ of the sum exactly when the first i monoid elements multiply to a

Defining addition, continued

- Carry computation corresponds to language $c^*(ac^*bc^*)^*$
- Definable in $FO^2[Bet]$, using the sentence:
  $\forall x : (b(x) \supset \exists y < x : a(y)) \wedge (a(x) \supset \exists y > x : b(y)) \wedge$
  $\forall y > x : (a(x) \wedge a(y) \supset b(x, y)) \wedge (b(x) \wedge b(y) \supset a(x, y)))$
- Hence the language ADD is also definable in $FO^2[Bet]$
- On the other hand the betweenness relations are definable in $FO^2[<, Add]$ using the binary relation $Add(x, y)$, where the positions from x to y (both included) represent vertically the three numbers $\begin{pmatrix} m \\ n \\ m+n \end{pmatrix}$

Defining circuits

- With alphabet $\{0, 1, or_1, and_2, or_3, \ldots, and_{2k}, or_{2k+1}, \ldots\}$ one can encode in prefix form constant-depth boolean circuits with inputs set to 0 and 1
- $CIRC_1 = or_1(0+1)^*(0+1)$ encoding circuits of depth 1, and
- $TRUE_1 = or_1(0+1)^*1(0+1)^*$ encoding such circuits that evaluate to *true*.
- $CIRC_2 = and_2(CIRC_1)^* CIRC_1$ for the next level,
- $TRUE_2 = and_2(TRUE_1)^* TRUE_1$ for the circuits evaluating to *true*, and so on.

Theorem

$FO^2[Bet]$ intersects every level of the quantifier alternation and AC^0 hierarchies (with growing alphabet).

Defining circuits

- With alphabet $\{0, 1, or_1, and_2, or_3, \ldots, and_{2k}, or_{2k+1}, \ldots\}$ one can encode, with one lookahead, prefix form constant-depth boolean circuits (here depth 3)
- First level: $CIRC_1(x, y) = or_1\{0,1\}\{0,1\}^*\{and_2, or_1, \triangleleft\}$,
  $BAD_1(x, y) = or_1\{and_2, or_1, \triangleleft\}$ (gates without inputs),
  $TRUE_1(x, y) = CIRC_1(x, y) \cap A^*1A^*(x)$,
  $FALSE_1(x, y) = CIRC_1(x, y) \cap \overline{A^*1A^*}(x)$
- Second level:
  $CIRC_2(x, y) = and_2 or_1\{0, 1, or_1\}^*\{and_2, \triangleleft\} \cap \overline{A^* BAD_1 A^*}(x)$,
  $BAD_2(x, y) = and_2 or_1\{0, 1, or_1\}^*\{and_2, \triangleleft\}, \cap A^* BAD_1 A^*(x)$
  $FALSE_2(x, y) = CIRC_2(x, y) \cap A^* FALSE_1 A^*(x)$,
  $TRUE_2(x, y) = CIRC_2(x, y) \cap \overline{A^* FALSE_1 A^*}(x)$
- Third level:
  $CIRC_3(x, y) = or_3 and_2\{0, 1, or_1, and_2\}^* \triangleleft \cap \overline{A^* BAD_2 A^*}(x)$,
  $TRUE_3(x, y) = CIRC_3(x, y) \cap A^* TRUE_2 A^*(x)$,
  $FALSE_3(x, y) = CIRC_3(x, y) \cap \overline{A^* TRUE_2 A^*}(x)$

Definability

- Regular languages of words are precisely those defined by sentences of a monadic second-order logic (Büchi-Elgot-Trakhtenbrot)
- Regular languages can be characterized by varieties of finite monoids (Eilenberg)
- A syntactic monoid corresponds to a minimal automaton for a language (Myhill-Nerode)
- $FO[<]$ corresponds to the variety of aperiodic monoids where for some n, for every element x we have $x^n = x^{n+1}$ (Schützenberger)
- For example, with $n = 10$: $xxxxxxxxxx = xxxxxxxxxxx$
- Intuitively, first-order logic cannot count beyond a point

Two-variable definability (Schützenberger, Thérien-Wilke)

- Consider an idempotent element $e = ee$
- M_e is the submonoid generated by factors of e, that is, generated by $\{f \mid e = pfq, \text{for some } p, q\}$
- A monoid is in variety **DA** iff for all idempotents e, $eM_e e = e$
- Equivalently, any J-class with an idempotent must only have idempotents
- $FO^2[<]$ corresponds to the variety of monoids **DA**
- For syntactic monoids M of languages in $FO^2[<, Suc]$, for every idempotent $e \neq 1$, eMe in **DA**

Non-definability argument

- We know that BB1 = $(ab)^*$ is not definable in $FO^2[<]$
- The language $c^*(ac^*bc^*)^*$, where c is called a neutral letter, is not definable in $FO^2[<, Suc]$
- Intuitively successor predicates can only affect local neighbourhoods
- The logic $FO^2[<, Suc]$ is no better at defining $c^*(ac^*bc^*)^*$ than $FO^2[<]$ is at defining $(ab)^*$ (Crane Beach property)
- For BB2 = $(ab)^*(a(ab)^*b(ab)^*)^*$, intuitively threshold constraints cannot keep track of differences in occurrences of letters over many occurrences
- Can we make this formal?

Necessary condition for definability

Theorem

*The syntactic monoid of a language definable in $FO^2[Bet]$ satisfies, for every idempotent e in the monoid, that the submonoid eM_ee is in **DA**.*

Proof.

Extends (Thérien-Wilke) game arguments for $FO^2[<]$. □

Two-pebble EF game for $FO^2[Bet]$: when Spoiler makes a move on one word responded to by Duplicator in the other word, the set of letters between the two pebbles and the letters at them have to be the same in both words

Summary

- Over linear orders, (Kamp) showed that binary modalities capture FO^3 properties like betweenness, and this is sufficient for all of first-order logic
- LTL has low processing complexity, which makes it very successful as a description language
- We have several two-variable logics with letter-counting relations and matching temporal logics which are above $FO^2[<, Suc]$, but below $FO[<]$, in fact below a smaller algebraic variety
- The logics intersect the quantifier alternation hierarchy of FO, the until-since hierarchy of LTL, the depth hierarchy of AC^0 at all levels, but do not include all levels
- (Gabbay) Over binary relational structures (graphs) (D, R), no finite-variable fragment captures first-order logic

Towards a descriptive *FO*

- There is a two-hop journey from Chennai to Seoul whose fare is cheaper than the one-hop journey
- Going from India to Brazil via Europe or the US takes more than a day, there is a shorter connection via Dubai and no visa hassles
- For a week's visit, you may prefer to stay in an apartment (rather than a hotel), buy groceries and cook
- Useful description logics already incorporate threshold counting features beyond two-variable logic, nominals (used for parts of a graph above), and transitive closure beyond first-order logic