Course Script

IN 5110: Specification and Verification of Parallel Systems

IN5110, autumn 2019

Martin Steffen, Volker Stolz

Contents

Chapter 4
μ-calculus model checking

What is it about?

Learning Targets of this Chapter

The chapter covers an short intro to the (resp. one variant) of the μ-calculus and model-checking it. We focus on the most prominent version of the μ-calculus for model checking known as modal μ-calculus, with a "branching time" interpretation. The logic can be understood as the "prototypical" **logic with fixpoints**, so we'll have to talk about fixpoints, as well. For model checking, we look at a bit of "game theory" (parity games).

Contents

4.1 Introduction

The presentation takes information from the handbook article Bradfield and Walukiewicz [1], but cannot cover all of the theoretical background in there (and there's a lot, due to the rather fundamental nature of the μ-calculus). There are many starting points that have led to what here is called μ-calculus. In computer science, one important reference point is the article Kozen [3].

Intro remarks

- rather fundamental logic
- central to μ-calculus: **fixpoints**
- many variations (and names)
  - propositional μ-calculus
  - modal μ-calculus
  - Hennessy-Milner logic with recursion
  - ....

For the lecture: vanilla μ-calculus a plain, propositional modal logic + fixpoints

It's not an exaggeration to say, "the" μ-calculus is "fixpoint logic" as it's all about fixpoints. Depending on the starting point, one can (and some did) add fixpoints to, for example, first-order logic and what not. For the model checking lecture, we take as a starting point a *modal logic* (like the ones we discussed in the respective chapter). Actually, we are going for a multi-modal logic. The intuition behind the logic is that the modalities talk about transitions (labelled transitions in fact, as we have a multi-modal logics), not about "knowlege" or "beliefs" etc. One can for sure also think of adding fixpoints with such more philosophical interpretations in mind. However, we focus on Kripke structures resp. transitition system representing steps in the executions of programs or systems. Like we mostly did for LTL (and CTL etc. covered by student representation), the starting point is also a *propositional* core logic, underneath the modal part (but that also is a bit orthogonal: there will be a student talk about QTL, quantified temporal logic, which allows first-order quantification. One could also turn that into a μ-calculus-like formalism). This set-up here is kind of the vanilla propositional μ-calculus. We start by recalling what, very generally, a fix-point of a function is. That's actually pretty simple.

What's a fixpoint?

$f : A \to A$

$$f(a) = a$$

A fixpoint of a function f is is thus defined quite simple (the a from set A in the above example is a fixpoint if f). The next part of the lecture is more a warm-up, as a reminder that there are actually fixpoints "everywhere". It's not always explicitly stated, like "let such-and-such be defined as fixpoint in the following way ...", there are other formulations used, and we have encountered such formulations in the lecture already (perhaps without being aware that underlying the respective definition, there was actually a fixpoint construction. The definitions were all on the "meta-level" not as part of a logic, i.e., fixpoints were mostly used (implicitly) to define or talk *about* a logic, we did *not* introduce fixpoints as part of the logics (that what the μ-calculusdoes). Actually, it's not 100% correct when saying that so far the logics did not allow to express fixpoints. It will turn out that temporal operators such as "eventually", "always", "until" are effectively fixpoint constructors. Only that there are no explicit fix-point constructors, so their nature as fixpoints is hidden.

Fixpoints are everywhere

A pedestrian definition of syntax The set Φ of *propositional formulas* is given as follows

- all propositional constants from AP are formulas
- if φ is a formula, then so is $\neg\varphi$
- if φ_1 is a formula and φ_2 is a formula, then so is $\varphi_1 \wedge \varphi_2$
- if φ_1 is a formula and φ_2 is a formula, then so is $\varphi_1 \vee \varphi_2$
- ... [more constructs if wished] ...

Is that even a definition?

Fixpoints are everywhere

A pedestrian definition of syntax (reformulated) The set Φ of *propositional formulas* is given as follows

- $AP \subseteq \Phi$
- if $\varphi \in \Phi$, the $\neg\varphi \in \Phi$
- if $\varphi_1 \in \Phi$ and $\varphi_2 \in \Phi$, then $\varphi_1 \wedge \varphi_2 \in \Phi$
- if $\varphi_1 \in \Phi$ and $\varphi_2 \in \Phi$, then $\varphi_1 \vee \varphi_2 \in \Phi$
- ... [more constructs if wished] ...

What about that?

$$\Phi = \{p, q, ..., p \wedge p, p \wedge q, p \wedge (p \vee q) \ldots\} \cup \{5, p\#q, \neg 5, 5 \wedge (q\#q) \ldots\}$$

The point of that example is that the set Φ of formulas given at the end *satisfies* the conditions, but it's not the one one had in mind. Basically, it means, the sentences as given are *not* a definition: they don't precisely *fix* the set of formulas, they just spell out conditions or constraints on Φ. Those constraints corresponds to **closure conditions**. For a condition like "if it so happens that $\varphi \in \Phi$ for some φ, then it's neccessary that also $\neg\varphi$ in Φ", one says more shorter that the set of formulas is **closed** under $\neg$, which is an *unary* operation or constructor.

How to fix(-point) it?

Depending to the style of writing or conventions in the field, one finds different ways of removing this ambiguity.

- ... **No other** entities are formulas, i.e. elements of Φ
- Φ is the **smallest set** such that 1) $AP \subseteq \Phi$, 2) ...
- Φ is **inductively** given by the following conditions: 1) ...

"Mu" $F(S) = AP \cup \{\neg\phi \mid \phi \in S\} \cup \{\varphi_1 \wedge \varphi_1 \mid \varphi_1 \in S, \varphi_2 \in S\} \cup \ldots$

$$\Phi = \mu F$$

The last "fix", the one labelled "Mu", approaches a definition based on explicit fixpoints. A function F is defined, which takes a set (the formal parameter S) and produces another set, by adding more elements (actually formulas). The the intended set Φ is given as the smallest fixpoint of that function F, i.e., the smallest set Φ such that $\Phi = F(\Phi)$. The traditional symbol that represents the smallest fixpoint is μ, i.e., $\Phi = \mu F$. As a side remark: the "mu" is a reference to some very famous popular science book GEB (Gödel, Escher, Bach. An Eternal Golden Braid), which in an deep, broad, and entertaining

way builds connections between art and logic, and where recursion (i.e., fixpoints) play a central role, and the book also contains something called MU-puzzle).

Fixpoints are everywhere, indeed

- grammars (with special syntax)

$$\varphi ::= AP \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \dots$$

- Kleene star and regular expressions:
  - finite words over Σ: written Σ^*
  - $a(b+c)^*$
- semantics of programming language
  - `while`-loop: make single steps, until termination (but not more!)
- data structures
  - the natural numbers are given ("constructed") by 0 and *succ* as constructor (and not more!)
- proof (and proof trees): a proof is given ("constructed") from *axioms* and application of *rules* (but not more!)

The list contains some examples which can be interpreted as involving fixpoints. A common denominator of the list is: all of them correspond to *smallest* fixpoints; it always about "the smallest collection closed unter something". The μ-calculus later allows two kinds of fixpoints, smallest ones with "μ" as binder and largest fixpoints ("ν"). It's probably fair to say, that one is used more to smallest fixpoints constructions, generally seen: One is very much used to list as data structure, which corresponds to a smallest fix-point. In some sense (that can be made precise, but is not relevant for the lecture), a correspondent largest fix-point data structure is their "infinite" counterpart, i.e., *streams*. As said, it's fair to say, that lists are a more common (and simpler?) data structure than streams. As a side remark, as came up during the lecture: infinite data structures, such as streams, can be seen as "lazy" data structure (as in Haskell). Lazy refers to the fact that, obviously, an infinite list (= stream) cannot be kept in memory in its infinite entirety, i.e., treated eagerly. What can be done is treating such an "infinite list" lazily, i.e., handing out the head of the last resp. the rest or tail of the streap piecewise and on demand.

Thinking about those kinds of examples (lists vs. streams, proofs as finite derivation trees vs. infinite proofs using infinite trees?...), the smallest fix-point indeed seems more common and less scary. In the μ-calculus later, however, the two fixpoints are completely symmetric. Actually, they are duals in a technical sense (like "$\neg\mu\neg = \nu$" and vice versa). What will be analogous is that smallest fixpoint formulas intuitively talk about *finite* "iterations" and largest fixpoints about *infinite* ones (we will see examples for that). Since the situation is symmetric in that sense, it's also misleading to think that μ-fixpoint necessarily are easier to understand compared to ν)-fixpoints. Arguably, the simplest useful temporal properties one might want to model-check are *invariants*. In LTL-notation, one can write $\Box p$ (for "always p holds). The proposition p must always hold, in all infinity, and that may give a hint that it corresponds to an ν-fixpoint. The dual formula "eventually q" (or $\Diamond q$ in LTL) is not harder or easier to understand and corresponds dually to a smallest fixpoint: the satisfaction of q is requestion within a *finite* time!

Connection to induction

Previously, the presentation stressed that "fixpoints are everywhere" (by giving examples mostly of data structures and constructions involving smallest fixpoints). As mentioned in that context: in textbooks one not often finds an *explicit* mention of the fixpoints, often other "keywords" are used ("the smallest set such that ..."). One keyword in that context is: "inductively", like: the natural numbers are given inductively by the constant 0 and the unary function (symbol) *succ*. The use of the word "inductively" is no coincidence: inductively given structures are equipped with a proof principle called **induction**. One presumably is most used to that principle in the context of natural numbers, so much so that induction in that setting is also called *mathematical induction* or *natural induction*. The latter word just means induction over natural numbers. The rather grandiose qualification as being the principle of "mathematical" induction perhaps is meant to differentiate that from "philosophical" induction, the principle to draw conclusions from special cases or instances to a general case. By way of mentioning: the latter interpretation is also connected to what the machine learning people mostly understand when they talk about *induction*. In that kind of interpretation of induction, is often positioned as opposite to *deduction*.

Anyway, we are here only interested in the mathematical proof principle of induction which roughly says (in the case of the natural numbers):

> if you need to prove some propery **for all** natural numbers, then the way to go for it is:
>
> 1. prove that the property holds for 0, and prove that,
> 2. if it holds for n, that implies that it also holds for $n + 1$.

Note that there are *infinitely* many numbers, each of which is *finitely* constructed though. The two parts of that proof principle are known, of course, as the base case and the induction case or induction step, and the assumption in the induction step that the property holds for n (which is arbitrary) is also called induction hypothesis.

So far, so familiar (hopefully). Now, the slide contains one well-known and early "formalization" of natural numbers. It's known as Peano's axioms or postulates (or Peano-Dedekind). Depending on the source, there may be addition axioms (for instance, stating properties of the equality symbol "=). We are interested not in that part, we are interested in the natural numbers alone. Relevant for us are in particular axioms 1. and 2. from the slide. Axiom 3. and 4. are not really too relevant for us (they have to do with the fact that the historical axioms also formalized axioms about $=$ as part of the logic and being very explicit about that).

Anyway, the really interesting one is axiom number 5 (here listed in equation (4.1)). It's often mentioned with this epithet (the famous fifth axiom of Peano... Whether it was in his own original writings numbered as the fifth, I have not checked). What it effectively does is another way of stating that the natural numbers (closed under 0 and successor) is *the smallest set* closed under those. It does so slightly indirectly, referring to another S is, and it stipulates that *any other set* closed in 0 and successor must be the natural numbers already. Actually, reading the sentence carefully, it states that for each set S, closed under 0 and successor

$$\mathbb{N} \subseteq S$$

not $\mathbb{N} = S$. Effectivly, it does not make a difference (see a bit further down below), after we have discussed the formulation of Peano's axiom in the form of equation (4.2).

remember: one way of formulation "Φ is **inductively** given as follows ...

Natural numbers

1. 0 is a natural number
2. if n is a natural number, then so is $succ(n)$ (written also $n+1$)
3. $n+1 = m+1$, then $n = m$
4. there is no natural number n with $n+1 = 0$

Peano Nr. 5

If a set S contains 0 and is closed under successor, then **all** natural numbers are in S. (4.1)

Peano Nr. 5: natural induction

$$\forall\varphi.\varphi(0) \wedge (\forall n.\varphi(n) \rightarrow \varphi(n+1)) \rightarrow \forall n.\varphi(n)\ . \tag{4.2}$$

Peano's 5th axiom is also called the **induction axiom**. That is more clearly felt in the second formulation of that principle, the one from equation (4.2). Instead of talking about subsets S of $\mathbb{N}$, that one speaks about a property φ of natural numbers. Both views are equivalent: a subset of $\mathbb{N}$ can be interpreted as a property and vice versa. Like: the even numbers correspond to the property of "being even" and "being even" characterizes the subset of $\mathbb{N}$ containing the even numbers.

Before wrapping up the discussion of the slide with some conclusions, let's have a closer look at the first definition of Peano's 5th postulate from equation (4.1), which is insofar informal that it's an English sentence. Apart from that it's actually unambiguous and in that sense formal. Anyway, let's be more explicit, in particular trying to shed light on the notion of *closure.*

For that, let's define the following function (on sets)

$$F(S) = \{0\} + succ(S) \tag{4.3}$$

It's meant as follows: Take set S and build $succ(S)$ (which is defined as $\{succ(s) \mid s \in S\}$) and add 0 to the resulting set (the + stands for "disjoint union", i.e., $\cup$ in case there are no common elements). As an example: If $S = \{\bullet, 6, a, 0\}$, then $F(S) = \{0, succ(\bullet), succ(6), succ(a), succ(0)\}$. The 0 and *succ* are best thought of as constructurs (i.e., "syntax"), so $succ(6)$ is just the constructor *succ* applied not to the element 6. We have no reason to say $succ(0)$ is 1, thought that might be our intention (and then: what is $succ(\bullet)$ supposed to mean, anyhow), Of course arabic numerals in a decimal positional system are a well-suited *notation* or convention to refer to natural numbers, but

that requires extra overhead, here we are concerned about "what are natural numbers" not so much "what's an efficient way of writing then and working with them". Now, the English formulation of (4.1) formulates (about the mentioned S) that S contains 0 and is closed under successor. One can also see that as as requiring that S is closed under the nullary constructor 0 (which is a "constant" symbol) and the unary constructor of successor. With this alternative (but equivalent) wording in mind, we can write the postulate from (4.1) a bit more formulaic as follows. A set S is *closed under* F, if

$$F(S) \subseteq S \tag{4.4}$$

Note that S plays the role of a "variable" in the condition or constraint from equation (4.4), as the sub-set "equation" formulates a condition on S. Furthermore, equation (4.4) is almost an equation: it almost requires $F(S) = S$, but not quite. If one had this equation instead of the in-equation or subset requirement from (4.4), one has another example of a **fixpoint**: $F(S) = S$ means that S has to be a fixpoint of S. The slight weakening for the closure formulation from equation (4.4), specifies "almost" fixpoint, but not quite. Solutions of such subset inequations in the form of (4.4) are called **pre-fixpoints**, so an S, satisfying the requirement is a pre-fixpoint of F. In other words: being a pre-fixpoint of F and being closed under F means exactly the same.

Now, as we illustrated with the "grammar example" and the "natural numbers" example: being closed is not good enough for **definining** uniquely the set; what is missing is the "nothing else" part or "the smallest such set" part. So we are after the smallest set S satisfying the closure condition of (4.4), resp. *the smallest pre-fixpoint of* F. It's obvious that every fixpoint is also a pre-fixpoint. The convers typically does not hold, there are strictly more pre-fixpoints than fixpoints (the $\subseteq$ condition is just weaker than requiring $=$). That's easy enough, and not too interesting. However, we are actually not interested in fixpoints or pre-fixpoints as such. We are interested in the *smallest* such things. Then, what *is* interesting and relevant indeed is the following: under standard circumstances there is **no difference** between **smallest fixpoints** and **smallest pre-fixpoints** for a given F. In fact, not only that, but there is exactly *one* unique smallest fix-point which at the same is the *one* unique smallest pre-fixpoint. This fact, which is, with a bit of background, straightforward to establish also justifies that one speaks (under standard circumstances) of **the** smallest or **the** least fixpoint (as opposed to some or other "miminal" fixpoint).

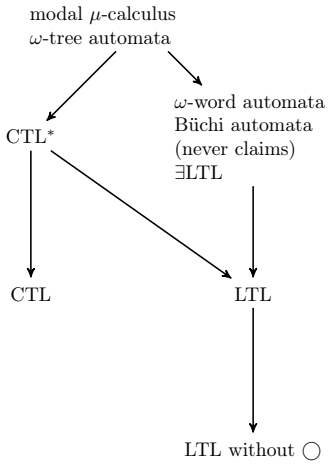
Now, as argued, the two formulations of Peano's 5th axiom, the ones from equations (4.1) and (4.2) are equivalent and also correspond to a (smallest) fixpoint requirement and, equivalently, allow the proof principle of mathematical induction. Looking especially at the formulation (4.2) where it's more visible: the formula is **not** a formula of first-order logic: there is quantification over "all propositions" φ in (4.2). Whatever logic it is, it's **NOT** first-order. One actually should tread rather careful here, it's slippery ground. The natural impulse could be: Oh, well, quantifying over propositions, I know already, that's *second-order logic*. Now, what may act as a warning sign here pondering the question of whether the above formula corresponds to a proposition itself. If so, one can have a logic where the quantifiers quantify over propositions *including the quantified formula itself* (the technical term in that connection is *impredicativity*). That kind of self-referential quantification may or may not lead to a catastrophe (catastophe in the sense that the

whole logic collapses and everything can be proven in that logic, and that mean really everything, including the negation of everything). In other words, the logic could become *inconsistent*. That form if self-referantiality is connected to the concept of whether there is a "set of all sets". If allows such things, one can introduce contradictions into set-theory (like the set of all sets that don't contain themselves).

Let's leave it at that. The importance for us is: Peanos induction axiom corresponds to a fix-point operator, and the discussion about the dangers of internalizing induction resp fixpoint into some logic should be an indication that this yields are some fairly powerful mechanism and it supports the status of the μ-calculusas an important logic.

A final remark: the discussion here centered on "natural induction", i.e., over $\mathbb{N}$, but completely analogous principles hold for other inductively given structures. In that case, one also speaks of *stuctural induction* (induction over the structure) and $\mathbb{N}$ is just a special case of a particularly simple structure.

Expressivity



The picture (shown earlier as well) shows again the order of different logics that are relevant in connection with model checking (and some automata model) with the μ-calculussitting on top. There is also a corresponding automaton model which we will not cover in the lecture.

4.2 Propositional μ-calculus: syntax and semantics

4.2.1 Syntax

Labelled transition systems

A *transition system* is a tuple

$$(S, \rightarrow, \{P_i\}_{i \in \mathbb{N}})$$

- $AP = \{p_0, p_1, p_2, \ldots\}$
- Act: $(= \Sigma)$ *actions* $a, b', \ldots$
- $\rightarrow \subseteq S \times Act \times S$
  - $s \xrightarrow{a} s'$, a-transition, from s to s'
- $P_i \subseteq S$
- note: switch of perspective for "proposition labelling"

Basically, it's a recap of earlier notions. The notion of (labelled) transition system here corresponds to that what we had before called Kripke model, except that now also the transitions are labelled by letters from an alphabet. Basically we encountered them in connection with multi-modal logics already. That's not surprising, insofar the flavor of the μ-calculuswe present here is bascially adding fixpoints to a multi-modal logic. In [1], they call the "models" just *transition systems* (not labelled transition systems), also what we introduced as alphabet Σ, a finite set of "symbols" or letters, they call that *action set*, the element of which are consequently called *actions*. Again, that's just terminology.

Syntax

φ	$::=$	$p \mid \neg p$		props and their negation
	$\mid$	X		variables
	$\mid$	$\varphi \wedge \varphi$	$\mid \varphi \vee \varphi$	2 usual boolean connectives
	$\mid$	$[a]\varphi$	$\mid \langle a \rangle \varphi$	(multi)-modal operators
	$\mid$	$\mu X.\varphi$	$\mid \nu X.\varphi$	fix-points

- *true* and *false*: $p \wedge \neg p$ resp. $p \vee \neg p$
- variables $X, Y \ldots \in Var$
- actions $a, b' \ldots \in Act$

Remarks on the syntax

- general negation: missing
  - especially $\neg X$ not part of the syntax
  - but: $\neg\varphi$ *definable*
- μ and ν (or σ when unspecific): **binding** operators
  - free and bound occurrences of variables
  - renaming of bound variables (α-renaming)

- some well-formedness conditions
  - don't reuse variables
  - guardedness

We don't go into details about those well-formedness conditions (they are important for some technical developments, only). But the concept of free and bound occurrences of variables should be familiar. Guardedness, just to given an impression, wants to avoid "immediate recursions" like $\mu X.X$ or also $\nu X.X \vee p$ (these are examples where X occurs unguarded). Those formulas are not contradictory, they have a meaning, but one tries to keep them out as they make some troubles in exstablishing results. Fortunately, some have proved that one can ignore them (insofar that one can always transform a formula into an equivalent one that avoids such situations; that particular transformation is not obvious, though...). Anyway, whereas $\mu X.X$ is not meaningless per se (just not nice to handle in establishing some results), formulas like $\mu X.\neg X$ **are meaningless** in that it cannot be given an interpretation in a manner that "makes sense". The cause of the problem is, that the body of the fixpoint construction, the formula $\neg X$ in that case, does not represent a *monotone* function. By **monotone** we understand *monotoneously increasing*.

During the lecture there had been comments that in some fields, the terminology of mononote functions capture both monotoneously increasing and monotoneously decreasing functions. For us, monotone refers to the increasing case (or at least not-decreasing), only. The "decreasing" situation, we would call *antitone*. Now, fixpoints are guaranteed to exists only in case the involved function is monotone, and that's the same for μ and for ν (it's not like: for μ, the function need to be mononote, for ν, on the other hand, antitone). Non-monotonic functions are out of the picture for us. That's why the syntax does not allow free-form negation, as there would be a danger to write formulas that act non-monotone. The syntax as given results in monotone functions, only. The (small) price to pay is that the syntax becomes slighty less compact than it would be otherwise: for each constructor, one also has to include its dual.

What about the variables?

- propositional μ-calculus
- $X, Y \ldots$ *different* from first-order logic variables
- variables here represent
  - formulas (from μ-calculus), resp.
  - semantically: *sets of states*

$\Rightarrow$ **second-order** flavor!

By "second-order" flavor we mean that we have variables that represent formulas (resp. sets of states captured by formulas). And we also (in some way) "quantify" over them, only not by existential or universal quantification $\exists$ and $\forall$, but by the fix-point binders μ and ν. At any rate, the variables $X, Y' \ldots$ are of quite a different nature from a setting if we had not the propositional μ-calculus, but had a first-order logic setting as underlying logic (with term variables x, y'). We see that also in the notion of *valuation* (when discussing the semantics). A valuation (or variable assignment) maps values or "semantics" to variables. Now, valuations will map variables from *Var* to set of states.

Preview on the semantics

- given transition system $\mathcal{M}$

Satisfaction relation

$$s \models_{\mathcal{M}} \varphi$$

$$s \models^{\mathcal{V}}_{\mathcal{M}} \varphi \tag{4.5}$$

Equivalently: semantic function semantics of φ in transition system $\mathcal{M}$ and with valuation $\mathcal{V}$:

$$[\![\varphi]\!]^{\mathcal{M}}_{\mathcal{V}} \in 2^S \tag{4.6}$$

$$\mathcal{V} : \mathit{Var} \to 2^S$$

There is not much "new" on the slides, compared to what we had in earlier chapters. Instead of the satisfaction relation $\models$, we define the semantics as "semantic function", traditionally written like $[\![\_]\!]$. That's just more convenient when later talking about fixpoints. The fixpoints will be built over functions (as usual), and therefore, writing down the semantics using the "semantical function" formulation comes in more handy.

Formulas are interpreted over a given transition system $\mathcal{M}$, i.e., the semantics of a formula φ is a set of states in that given $\mathcal{M}$. Since formulas may contain variables, the interpretation is additionally relative to choosing values for those free variables. So we assume a *valuation* $\mathcal{V}$ for those, with $\mathcal{V} : \mathit{Var} \to 2^S$, which assigns "semantics" (= sets of states) to each variable. That's why the semantic function from (4.6) or the satisfaction relation from (4.5) mentions $\mathcal{V}$ (besides $\mathcal{M}$).

Semantics (no fix-points)

The semantics for the logic without fixpoints is rather straightforward. That fragment is sometimes also called Hennessy-Milner logic (without recursion). It's basically the multimodal logic we have encountered earlier (in preparation of dynamic logic.)

$$\varphi ::= p_i \mid \neg p_i \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid [a]\varphi \mid \langle a \rangle \varphi$$

$$\begin{array}{rclrcl}
[\![\mathit{true}]\!]^{\mathcal{M}}_{\mathcal{V}} & = & S & [\![\mathit{false}]\!]^{\mathcal{M}}_{\mathcal{V}} & = & \emptyset \\
[\![p_i]\!]^{\mathcal{M}}_{\mathcal{V}} & = & P_i & [\![\neg p_i]\!]^{\mathcal{M}}_{\mathcal{V}} & = & S - P_i \\
[\![\varphi_1 \wedge \varphi_2]\!]^{\mathcal{M}}_{\mathcal{V}} & = & [\![\varphi_1]\!]^{\mathcal{M}}_{\mathcal{V}} \cap [\![\varphi_2]\!]^{\mathcal{M}}_{\mathcal{V}} & [\![\varphi_1 \vee \varphi_2]\!]^{\mathcal{M}}_{\mathcal{V}} & = & [\![\varphi_1]\!]^{\mathcal{M}}_{\mathcal{V}} \cup [\![\varphi_2]\!]^{\mathcal{M}}_{\mathcal{V}} \\
[\![[a]\varphi]\!]^{\mathcal{M}}_{\mathcal{V}} & = & \{s \in S \mid \forall s'.s \xrightarrow{a} s' \Rightarrow s' \in [\![\varphi]\!]^{\mathcal{M}}_{\mathcal{V}}\} & & & \\
[\![\langle a \rangle \varphi]\!]^{\mathcal{M}}_{\mathcal{V}} & = & \{s \in S \mid \exists s'.s \xrightarrow{a} s' \ \wedge \ s' \in [\![\varphi]\!]^{\mathcal{M}}_{\mathcal{V}}\} & & &
\end{array}$$

Fixpoints in LTL

- $\Box p$?
- $\Diamond p$
- $p\ U\ q$

Earlier, we made a big story around the fact that there are "fixpoints everywhere". Examples where mostly drawn from some constructions in "math" but not as integral part of a logic. Of course, the μ-calculusis kind of a the prototypical logic which offers fixpoints inside the logic, not just externally on the meta-level. But what about the other temporal logic we covered, LTL? On the next slide, we have a look at, for example, the "always" operator, $\Box$.

Reconsider for instance $\Box p$?

- fix-point equation for "always"?

$$\Box p = p \wedge \bigcirc \Box p. \tag{4.7}$$

choose $\Box p = \mathit{false}$

$$\mathit{false} = p \wedge \bigcirc \mathit{false} \tag{4.8}$$

Let's take $\Box$ for illustration. We are currently using LTL, which is a linear logic. The μ-calculusis in its general form a branching logic, but the arugments or illustrations work equally for a linear logic. Actually, one can also interpret the μ-calculusover linear transitions sytems, only. Anyway, "always p" can be intutively explained as

> "always p" means p now and, after one step "always p".

That's of course a recursive "definition" in that the thing one tries to define, "always p" is define in terms of itself (we should thought be careful with the word definition, as there may be more than one solution to that equation; and indeed there are). It also corresponds to the *unrolling* feature of $\Box p$, as given in equation (4.7). Seeing it this way means, we have been given $\Box p$ already. The chapter about LTL gave a definition of the semantics of $\Box p$; maybe it was given indirectly, in that we focused on U alone, but anyway, it was somehow defined, and the definition would state that a path satisfies $\Box p$, if $\forall i$, the suffix of the path starting at i satisfies p. So far, so familiar. Point being, equation (4.7) states a fact about $\Box p$ whose definition is given already.

However, we can try to use the equation also as **defining** $\Box p$. In that case we should probably write better

$$\text{"}\Box p\text{"} = p \wedge \bigcirc \text{"}\Box p\text{"}\ ,$$

to make clear that "$\Box p$" is something that we want to define and which we intend to satisfy the stated (recursive) equation. Since we want to "solve" that equation (to define $\Box p$), it's probably even better to write

$$X = p \wedge \bigcirc X \tag{4.9}$$

i.e., to use a variable X (as opposed to "$\Box p$") to represent the (yet to define) thing for which we want the equation to hold. That, of course, can be seen as a fix-point equation (in the discussion here, we dispense with the distinction of fixpoints (using $=$) and pre-fixpoints (using $\subseteq$ or implication).

Now, we can try to solve it. We have heared about the importance of smallest fix-points extensively. The smallest imaginable solution (in terms of sets of states) would be the empty set. That corresponds alternatively to the proposition *false*. So, let's fill in *false* for the unknown X (or interpreting semantically X as $\emptyset$), checking if $\mathit{false} =^{?} p \wedge \bigcirc \mathit{false}$ (equation (4.8) from the slide). Well, $\bigcirc \mathit{false}$ is equal to *false*, which means *false* is indeed a solution to the fixpoint equation (4.9). And, being the empty set, it's definitely the *smallest* fixpoint.

That's a disappoinment, since we set out to give a fixpoint definition of $\Box p$, not of *false*. Now, since the smallest fixpoint was a failure, the only hope is the largest fixpoint (nobody needs fixpoints somewhere in the middle...) As it turns out, the largest fixpoint is what we are after, it corresponds to $\Box p$. So we can define

$$\Box p = \nu X.p \wedge \bigcirc X \ . \tag{4.10}$$

That's all fine and good, but one may feel uneasy, for instance by the obvious "failure" of $\mu X.p \wedge \bigcirc X$. Maybe in this case it's clear that it can't be the smallest fixpoint (as it's *false*), so it must in all plausibility the largest. But maybe there are situations where the smallest fixpoint does not collapse to *false* but makes some form of sense, it's only not the one we planned for. How do we know that the definition from equation (4.10) is what we want? At least it *is* a definition, because we know that in the given circumstances, the greatest fixpoint is unique (as is the smallest fixpoint which turned out to be *false*). We could leave it like that: we defined $\Box p$ by (4.10) and that just it. However, we already had an independent definition of "always p" not involving μ in an apparent way, and beside that we have an intuition what "always p" is supposed to mean. How can we make sure that our intention is captured by the definition from (4.10)? Or in general: what the intiution, when should one choose μ and when ν?

Now, it's tricky to communicate "intiution", intuition builds up by doing things and exercising and repeating things. After a while things seem "natural" and intuitively clear, and one forgets that one had been struggling with the concept in the beginning. That's also when it becomes hard to communicate the idea, because stating "it's intuitively clear that ..." is not helpful. But many things in math are like that and for many things we have forgotten that they may not seem clear at the beginning. Notions about "infinity" are notoriously "unclear" (and philosophically debated, like in which sense does the "infinite" exist, etc). In some sense we get used to it: $\mathbb{N} = \{0, \mathit{succ}(0), \mathit{succ}(\mathit{succ}(0))...\}$ (or more commonly $\{0, 1, 2, ...\}$, and no one looses sleep about the "...", we all know that there "are" infinitely many natural numbers (maybe on a computer only up to `MAXINT`, but, well, in priciple, you know...). We are just used to deal conceptually with infinite sets without breaking a swet (like natural numbers, the set of propositional formulas as give

by a grammar, the set Σ^* and those examples from before). It may be noted that even other mathematical facts needed time "to get used to" (historically). One is 0. There had been a time where people had scruples to "have" or work with 0, like: "how can there be or should there be there be a symbol for the absense of something". Zero stands for "nothing" and one cannot meaningfully compute with nothing (there was a time also where the people had the concept of 0 but did not use a symbol for it, like leaving the place empty. That did not help readability much. And it took time to "get used to it" and have the courage to write 0 for something that "actually" (?) does not exists. Same with the imaginary numbers (numbers like $\sqrt{-1}$ that do not really exist, or do they?)

The examples about inifinite sets ($\mathbb{N}$, Σ^* ...) correspond perhaps not coinicidentally to *smallest* fixpoints. It's about *infinite sets* build up by using finite constructions for each elements. $a^* = \{\epsilon, a, aa, aaa, \ldots\} = \{a^n \mid n \in \mathbb{N}\}$ which we understand as the set containing words using a, where the words are of **arbitrary length** n. Implicit in the word "arbitrary length" is that it not really "arbitrary" in the sense "n = infinite length". Not arbitrary in this radical sense, just ordinary arbitraryness please, like $n \in \mathbb{N}$. $\mathbb{N}$ is infinite, that's fine, but n please not (because infinite numbers don't really exist, or do they?...).

This discussion is meant less historical, but more of giving intution about μ and ν. The smallest fixpoints are of the form as in a^n with $n \in \mathbb{N}$. They may describe a infinite collection of entities (like words or numbers) each of which is finite. \$$\mu$\$-based definition corresponds to constructions where the single elements are conceptually *infinite* (or at least include infinite elements). One example we have actually seen was Σ^ω, the set of infinite words over Σ.

4.2.2 Background: Fixpoints

Orders, lattices, etc.

as a reminder:

- partial order $(L, \sqsubseteq)$
- *upper bound* l of $Y \subseteq L$:
- *least* upper bound (lub): $\bigsqcup Y$ (or *join*)
- dually: lower bounds and greatest lower bounds: $\bigsqcap Y$ (or *meet*
- **complete lattice** $L = (L, \sqsubseteq) = (L, \sqsubseteq, \bigsqcap, \bigsqcup, \bot, \top)$: a partially ordered set where meets and joins exist for *all subsets*, furthermore $\top = \bigsqcap \emptyset$ and $\bot = \bigsqcup \emptyset$.

There are also other forms of lattices, for instance, if one only needs joins, but not meets, one can get away with a semi-lattice, and there are many more variations. For the lecture, we generally simply assume *complete lattices* and thus, the montone framework is happy. In particular, if we are dealing with *finite* lattices, which is an important case, we don't need to consider *infinite* sets, and "standard" lattices with binary meets and joins (and least and largest elements) are complete already.

Fixpoints

given complete lattice L and monotone $f : L \to L$.

- **fixpoint:** $f(l) = l$

$$Fix(f) = \{l \mid f(l) = l\}$$

- f *reductive* at l, l is a **pre-fixpoint** of f: $f(l) \sqsubseteq l$:

$$Red(f) = \{l \mid f(l) \sqsubseteq l\}$$

- f *extensive* at l, l is a **post-fixpoint** of f: $f(l) \sqsupseteq l$:

$$Ext(f) = \{l \mid f(l) \sqsupseteq l\}$$

Define "lfp" / "gfp"

$$lfp(f) \triangleq \bigsqcap Fix(f) \quad \text{and} \quad gfp(f) \triangleq \bigsqcup Fix(f)$$

The last display just gives the names to the two elements of the lattice defined by the corresponding right-hand sides. We know that those elements are existing thanks to the fact that L is a complete lattice (and it's very easy to see that meets and joins are unique, that means the $lfp(f)$ and $gfpf$ are well-defined elements of the lattice). The chosen names somehow suggest that the two thusly defined elements are the least fixpoint, resp. the greatest fixpoint of the monotone function f.

But, so far *lfp* and *gfp* is just a suggestive choice of name. It requires a separate *argument* that the elements are actually fixpoints, and the least, resp. the largest fixpoint as that as well. Finally, if we take it really serious, an argument should be found that allows to speaking of *the* least fixpoint. If there is more than one least fixpoint, one should avoid talking about "the least fixpoint" (same for the largest fixpoint). The argument for uniquess of least fixpoints (or for greatest fixpoint) is very simple though, similar to arguing for the uniqueness of "the least upper bound" etc.

If one would carry out the argument, i.e., the proof, that all fits together in the sense that the $lfp(f)$ and $gfp(f)$ defined above *are actually* the least fixpoint and the largest fixpoint, and if one would carefully keep track of what is actually needed to make the proof go through step by step, then one would see that *every single condition* for being a complete lattice is needed (plus the fact that f is monotone). If one removes one condition, the argument fails! Conversely that means the following: We are interested in uniquely "best approximations" (least or greatest fixpoints depending in whether it's a may or a must analysis),

> and, having a monotone f, a complete lattices **is exactly what guarantees that those fixpoints exists**. Exactly that, nothing less and nothing more, If your framework has monotone functions and is based on a complete lattice, it works. If not, it does not work, very simple.

That explains the importance of lattices and monotone function. Also, I would guess that historically, the *need* to assure existance of fixpoints has led Tarski (the mathematician whose concepts we are currently covering) exactly to the definition of lattice, not the other way around ("oh, someone defined some lattice, let's see what I can find out about them, perhaps I could define some $lfp(f)$ like above and see if I could prove something iteresting about it, perhaps it's a fixpoint?". But as said, that is speculation.

Having stressed the importance of complete lattices, for fairness sake it should be said that there's also a place for analyses which fail to meet those conditions. In that case, one might not have a (unique) best solution. Perhaps even worse (and related to that), one might need *combinatorial* techniques (like backtracking), i.e., checking all possible solutions to find an acceptable one. If that happens, the *cost* of the analysis may explode. To avoid that one may give up to look for a "best solution" and settle for a "good enough" one and heuristics that hopefully find an acceptable one efficiently, or even throw the towel and give up "soundness". Anyway and fortunately, plenty of important analyses fit well into the monotone framework with its lattices, its unique best solution and —perhaps best of all– its efficient solving techniques. Therefore this lecture will cover only those here. Those are called classical *data flow analyses*.

Tarski's theorem

Core Perhaps core insight of the whole lattice/fixpoint business: not only does the $\bigsqcap$ of all pre-fixpoints uniquely exist (that's what the lattice is for), but —and that's the trick— *it's a pre-fixpoint* **itself** (ultimately due to montonicity of f).

Theorem 4.2.1. *L: complete lattice, $f : L \to L$ monotone.*

$$\begin{array}{lcll} lfp(f) & \triangleq & \bigsqcap Red(f) & \in \quad Fix(f) \\ gfp(f) & \triangleq & \bigsqcup Ext(f) & \in \quad Fix(f) \end{array} \tag{4.11}$$

- Note: *lfp* (despite the name) is *defined* as glb of all pre-fixpoints
- The theorem (more or less directly) implies *lfp* is the *least* fixpoint

Fixpoint iteration

- often: iterate, approximate least fixed point from below $(f^n(\bot))_n$:

$$\bot \sqsubseteq f(\bot) \sqsubseteq f^2(\bot) \sqsubseteq \dots$$

- not assured that we "reach" the fixpoint ("within" ω)

$$\bot \sqsubseteq f^n(\bot) \sqsubseteq \bigsqcup_n f^n(\bot) \quad \sqsubseteq \quad \begin{array}{l} lfp(f) \\ gfp(f) \quad \sqsubseteq \bigsqcap_n f^n(\top) \sqsubseteq f^n(\top) \sqsubseteq (\top) \end{array}$$

- additional requirement: **continuity** on f for all ascending chains $(l_n)_n$

$$f(\bigsqcup_n (l_n)) = \bigsqcup (f(l_n))$$

4.2.3 Semantics

Semantics of formulas with free variables

- apply the FP theorem (Knaster-Tarski)
- assume $\mu X.\varphi(X)$ or $\nu X.\varphi(X)$
- $\mathcal{M}$ with state set S: fixed
- consider semantics of body $\varphi(X)$,
  - assume (for simplicity), only one free variable X

$$[\![\varphi(X)]\!]^{\mathcal{M}} : 2^S \to 2^S$$

- 2 welcome facts
  1. 2^S a **complete lattice**
  2. the function is **monotone** (and also **continuous**, under reasonable assumptions)
- general case: φ may have more variables than just X

$$f(S') = [\![\varphi]\!]^{\mathcal{M}}_{\mathcal{V}[X \mapsto S']} : 2^S \to 2^S$$

with $S' \subseteq S$

Semantics of the fixpoints

$$\begin{aligned}
[\![\mu X.\varphi]\!]^{\mathcal{M}}_{\mathcal{V}} &= \bigcap \{S' \subseteq S \mid [\![\varphi]\!]^{\mathcal{M}}_{\mathcal{V}[X \mapsto S']} \subseteq S'\} && \text{(lfp)} \\
&= \bigcap \{S' \subseteq S \mid f(S') \subseteq S'\} \\
[\![\nu X.\varphi]\!]^{\mathcal{M}}_{\mathcal{V}} &= \bigsqcup \{S' \subseteq S \mid S' \subseteq [\![\varphi]\!]^{\mathcal{M}}_{\mathcal{V}[X \mapsto S']}\} && \text{(gfp)} \\
&= \bigsqcup \{S' \subseteq S \mid S' \subseteq f(S')\}
\end{aligned}$$

where $f(S') = [\![\varphi]\!]^{\mathcal{M}}_{\mathcal{V}[X \mapsto S']}$

Alternation of fixpoints

- expressivity of μ-calculus: due to fix-points
- more precisely: "nesting" of fix.points
- even more precisely: **alternation-depth of nested fixpoints**.
- compare: direct recursion vs. mutual recursion
- similarly: "$\mu^2 = \mu$"
- technical definition of nesting: not 100% immediate
  1. $(\mu X.\varphi) \wedge (\nu X.\varphi_2)$: no nesting
  2. $\mu X.\mu Y \varphi(X, Y)$: no alternation
  3. $\nu X.((\mu Y.p \vee \langle b \rangle Y) \wedge [a]X)$: ??
  4. ???

The definition is not 100% obvious. In particular, the formula number 3 is **not** nested in the way covered by the definition. Naively it is: the μ fixpoint is inside the ν fixpoint, so there seems to be nesting. Also there is alternation. But: as said, the definition is slightly more complex. The sitution in formula 3 does not count as proper nesting, in that the two "loops" are somehow "independent". The inner loop is *completely enclosed* in the outer loop. Why that may be called a form of nesting, but it's not the "proper" nesting that gives the μ-calculusits power.

4.3 Model checking

Game

Definition 4.3.1 (Game)**.** A game is a triple $G = (V, T, Acc)$ where

1. V are *nodes* partitioned between two players, Adam and Eve: $V = V_A + V_E$.
2. $T \subseteq V \times V$ is a *transition relation* determining the possible successors of each node, and
3. $Acc \subseteq V^\omega$ is a set defining the *winning condition*

- node: aka **position**
- Acc: *winning condition*

Playing a game like this

- two-player game
- two kinds of nodes (Eve's and Adam's)
- game "moves" through positions
  - in one of Eve's nodes: *Eve* **chooses**
  - analogous for Adam
- **winning**:
  - winning condition: from the perspective of Eve
    - ininite path through G: if Acc satisfied, Eve wins, otherwise Adam
  - a player "stuck": looses as well
  - no draw possible
- winning *node*: $\exists$ winning strategy

strategy θ (for Eve) Given G. For each sequence of nodes $\vec{v}$, ending in a node $v \in V_E$: *choose* $\theta(\vec{v}) = v'$, such that $v \to v'$

Games as general framework

- "game theory": broad field with many applications
- here: game used for
  - semantics, logics, model-checking
- situation often: open systems

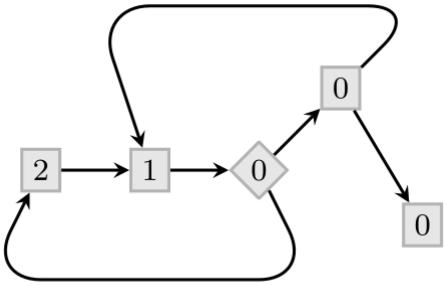
  - enviroment context $\leftrightarrow$ program
  - attacker $\leftrightarrow$ system
  - controllable $\leftrightarrow$ non-controllable parts

Game Different, players with own "goals" (conflicting or at least different) and local "influence" or control.

Rest

- many variations
  - 2-player, multi-player
  - zero-sum games (no win/win situation in those...)
  - restricted information
  - probabilistic ("mixed") strategies (Nash-equilibrium!)
  - ...

Game example



- nodes or positions in the graph
  - Eve: "diamond"-shaped
  - Adam: "box"-shaped
- winning condition (here): Eve wins, if the game passes through "2" infinitely many times
- numbers in the nodes: "ranks" (see later)

Eve (in this example) has only one node, where she can choose going up or down. If she chooses uniformely each time to go down, she wins, as she passes infinitely often through the node marked with 2. This condition is, in this example, the winning condition *Acc* (which is formulated from the perspective of Eve). If she goes up, Adam then get's a choice. He could actually prevent Eve from ever reaching 2 again, but in doing so, and going down, he will be stuck, and thereby loose immediately (getting stuck means loosing). Therefore, he is forced to take the loop back.

Note: even if Eve cannot control or know what Adam does, she can rely on the fact that he acts "rational" in the sense that he wants to make moves with the goal of winning (or not loosing)

BTW: not loosing is actually the same thing as winning since in the games we are looking at, the outcome is binary or "boolean": either Adam wins and Eve looses, or the other

way around. We use a game formulation to determine if a property holds in a state or not, and that is either the case or not, as in classical logic.

Positional strategies

- stategy in general $\theta(\vec{v}v) = v'$
- in the example: strategy of Eve: can be dependent on the "current node" only

$\Rightarrow$ **memoryless** or **positional**

$$\theta(v) = v'$$

Parity games

given game G

Parity winning condition **ranking**: $\Omega : V \to \{0, 1, \ldots, d\}$

$$Acc = \{\vec{v} \in V^\omega \mid \lim_{i \to \infty} \sup \Omega(v_i) \text{ is even}\}$$

Mostowski [4], Emerson and Jutla [2]

PWC theorem

- every position is winning for one of the two players
- it's winnable by a **positional** strategy
- it's **decidable** who wins

Model checking μ-calculus and parity games

Verification game

$$\mathcal{M}, s \models \varphi \quad \Rightarrow \quad \mathcal{G}(\mathcal{M}, \varphi)$$

$\mathcal{M}, s \models \varphi$ iff Eve wins from position corresponding to s and φ in $\mathcal{G}(\mathcal{M}, \varphi)$

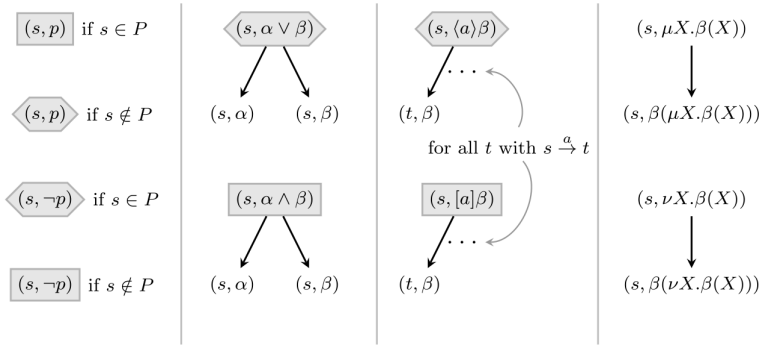
- $\mathcal{V}$: valuation for free vars, i.e., game $\mathcal{G}_\mathcal{V}(\mathcal{M}, \varphi)$
- **positions** in the game

$$(s, \psi)$$

ψ: 1 formula from the closure of φ

Intention of the construction Eve has winning strategy from (s, ψ) iff $\mathcal{M}, \mathcal{V}, s \models \psi$

Rules of the verification game



Ranking

- ranking positions with fp- formulas $\nu.\psi$ or $\mu.\psi$

1. μ-formula odd.
2. ν-formula even

Rest

- additionally: subformulas should must no higher ranks than the surrounding formula
- standard way: connect to **alternation depth**

Rest

$$\begin{array}{lcll} \Gamma(\psi) & = & 2\lfloor adepth(X)/2 \rfloor & \text{for } \psi = \nu X.\psi'(X) \\ \Gamma(\psi) & = & 2\lfloor adepth(X)/2 \rfloor + 1 & \text{for } \psi = \mu X.\psi'(X) \\ \Gamma(\psi) & = & 0 & \text{otherwise} \end{array}$$

Bibliography

[1] Bradfield, J. and Walukiewicz, I. (2018). The mu-calculus and model checking. In Clarke, E. C., Henzinger, T. A., Veith, H., and Bloem, R., editors, *Handbook of Model Checking*. Springer Verlag.

[2] Emerson, E. A. and Jutla, C. (1991). Tree automata, mu-calculus, and determinacy. In *IEEE Symposium on Foundations of Computer Science (FOCS'91)*.

[3] Kozen, D. (1983). Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354.

[4] Mostowski, A. W. (1984). Regular expressions for infinite trees and a standard form of automata. In Skowron, A., editor, *Computation Theory*, volume 208 of *Lecture Notes in Computer Science*, pages 157–168. Springer Verlag.

Index