Info: Video 4a,4b

Model checking in QBF

```
def Model-check(phi,S):
        #### same as in propositional logic
        if phi = p:
                return S(p)
        elif phi = phi_1 or phi_2:
                return Model-check(phi_1,S)
                                or Model-check(phi_2,s)
        elif phi = phi_1 and phi_2:
                return Model-check(phi_1,S)
                                and Model-check(phi_2,s)
        elif ...
         ####
        elif phi = \exists p phi':
                return Model-check(phi',S[p:= True])
                        or Model-check(phi',S[p:=False])
        elif phi = \forall p phi':
                return Model-check(phi',S[p:= True])
                        or Model-check(phi',S[p:=False])
```

The new added parts are setting the variable p to the wanted value and with the $\exists$ one of the two Model-checks need to be True with the $\forall$ both of the Model-checks need to be True. `phi`' denotes in the algorithm the part of the formula that is behind the $\exists$ and the $\forall$.

The time complexity is exponential as we have two recursive calls each $\exists$ or $\forall$. i.e. $O(|\phi|^{q \cdot d(\phi)})$ where q is the number of quantifiers, and $d(\phi)$ is the quantifier depth.

Note

what is the quantifier depth?
The number of consecutive quantifiers
$\exists p \forall q \forall r$ would have the quantifier depth 3

Complexity: **PSPACE**-complete

1 Satisfiability in QBF

Reduces the model-checking for QBF
QBF -> $\phi(p_1, p_2..)$ satisfiable $\iff$ $\exists p_1 \exists p_2 \ldots \phi$ holds over empty structure.

What does this mean:
We can rewrite every Satisfiability problem of every formula ϕ into another formula ϕ' using the $\exists$

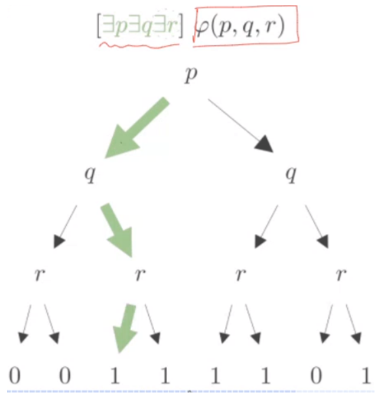
quantifier. Then we apply the model-checking on this new formula ϕ'. If there exists a solution to the formula the entire thing is satisfiable.i

This is caused by the meaning of the $\exists$ quantifier. $\exists$ asks is there one mapping onto either true or false i.e. is it satisfiable?

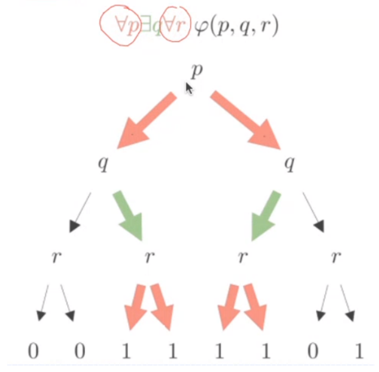
Complexity: **PSPACE**-complete

Now we look at the difference of the satisfiability tree in QBF and SAT (propositional logic)

Just has one path as there are only $\exists$ when looking at satisfiability



Has multiple paths as there are $\exists$ as well as $\forall$ which split up the tree



lemma 5

In QBF every formula is equivalent to one in Prenex Normal Form which can be computed efficiently

Example:
Not in Prenex Normal Form:
$\phi = (\exists p \phi_1) \wedge \phi_2$

Push quantifiers outside by using Lemma 4 (renaming):
$\exists p \phi_1 \vee \phi_2$ is rewritten into $\exists q(\phi_1[p/q]) \vee \phi 2$
$\exists p \phi_1 \wedge \phi_2$ is rewritten into $\exists q(\phi_1[p/q]) \wedge \phi 2$
...

One has to take care! the condition is that p is not occuring free in ϕ_2 otherwise we would change the meaning of the formula! Solution: one can rename the variable p in $\exists p \phi_1$ to another name i.e. u, l, k so that the variable does not occur freely in ϕ_2.

Lemma 6

Model-checking QBF's in Prenex Normal Form with n quantifiers ($\exists, \forall$) has the time complexity:

$$\Sigma_n = NP^{coNP^{\cdots}} \text{ or } \Pi_n = coNP^{NP^{\cdots}}$$

Application: the sequence of NP and co-NP depends on which quantors one uses: i.e. $\exists p \forall q \exists r \phi$ would have the complexity class $NP^{co-NP^{NP}}$

What does NP complexity mean:
The problem can be solved with a machine in a $non\ deterministic$ in polynomial time. When something is in the power i.e. NP^{coNP} it means that the main process has a subprocess that solves $coNP$ problems

Application:
problem: does the device admit a run from initial to target state? Simmilar as in the class Verification 3

Consider a device whose internal state is encoded by k bits $\underline{p} = p_1, \ldots, p_n$

- initial state is described by a propositional formula $\phi_{init}(\underline{p})$
- target state is described by a propositional formula $\phi_{target}(\underline{p})$
- Transitional formulas are described by a propositional formula $\phi_{step}(\underline{p}, \underline{p'})$

In Propositional Logic the solution went like this:
The formula describes it:

$$\phi_{reach}(\underline{p_1}, \ldots, \underline{p_n}) = \phi_{init}(\underline{p_1}) \wedge \phi_{step}(\underline{p_1}, \underline{p_2}) \wedge \ \ldots\ \wedge \phi_{step}(\underline{p_{n-2}}, \underline{p_{n-1}}) \wedge \phi_{target}(\underline{p_n})$$

Now we sum up all ϕ_{step} to get from $\underline{p_{init}}$ to $\underline{p_{target}}$ as $\phi_{run}[p_{init}, p_{target}]$.

In the next step we split the path up into two paths from p_init to p_r and from p_r to p_target with about equal length $\frac{n}{2}$. Then we split this sub-path again up recursively till the path is equal to $\phi_{step}[\underline{p_k}, \underline{p_{k+1}}]$ where k is some step in the path.
Mathematically written:

$$\phi^n_{run}(\underline{p}, \underline{q}) = \exists r(\phi^{n/2}_{run}(\underline{p}, \underline{r}) \wedge \phi^{n/2}_{run}(\underline{r}, \underline{q}))$$

$$\phi^{n/2}_{run}(\underline{p}, \underline{q}) = \exists r(\phi^{n/4}_{run}(\underline{p}, \underline{r}) \wedge \phi^{n/4}_{run}(\underline{r}, \underline{q}))$$

$$\ldots$$

$$\phi^1_{run}(\underline{p}, \underline{q}) = \phi_{step[\underline{p}, \underline{q}]}$$

The depth of this formulas is actually proportional to the number of bits k i.e. $O(2^k)$. The size of the formula should be of exponential size as in Propositional Logic

What is the advantage? Complexity is similar? We can re-write:

$$\phi^n_{run}(\underline{p}, \underline{q}) = \exists \underline{r} \forall \underline{s} \forall \underline{t}((\underline{s}, \underline{t} = \underline{p}, \underline{r} \implies \phi^{n/2}_{run}[\underline{p}, \underline{r}]) \wedge (\underline{s}, \underline{t} = \underline{r}, \underline{q} \implies \phi^{n/2}_{run}[\underline{r}, \underline{q}]))$$

This makes that we have two times $\phi^{n/2}_{run}(\underline{r}, \underline{q})$ that are equivalent and using Rules

$$(A \implies C) \wedge (B \implies C) \equiv (A \vee B) \implies C$$

We can rewrite again to:

$$\phi^n_{run}(\underline{p},\underline{q}) = \exists \underline{r} \forall \underline{s} \forall \underline{t} ((\underline{s},\underline{t} = \underline{p},\underline{r}) \vee (\underline{s},\underline{t} = \underline{r},\underline{q})) \implies \phi^{n/2}_{run}[\underline{r},\underline{q}]$$

With the result that the algorithm is linear in k

When using quantifiers we can make formulas much more succinct. And one can exchange a huge conjunction ($\wedge$) by a universal quantification

---

break

QBF

2 FO - First order logic

Vocabulary:

- Boolean connectives $\wedge, \vee, \neg, \iff, \implies$
- $\forall, \exists$ Quantifiers
- Relationship symbols $\Sigma = R, S, T$ (= always in the signature)
- first order variables: x, y, z

Syntax:

$$\phi = |\phi \wedge \phi | \phi \vee \phi | \neg \phi | \phi \iff \phi | \phi \implies \phi | \ldots | R(x_1 \ldots x_k) | \ldots | \exists x \phi | \forall x \phi$$

semantics :
A structure now consists of a *universe* U^s which is a set of objects

- *interpretations* $R \implies R^S \subseteq U^S \times \ldots \times U^S$
  $x \implies x^S \in U^S$
  What does this mean?
  The first R is the Relationship symbol. R^S is the real Relationship
  x is a variable that is getting mapped on an object of the universe x^s.

$$S \models \phi_1 \vee \phi_2 \iff S \models \phi_1 or S \models \phi_2$$

$$S \models R(x_1, \ldots, x_k) \iff (x^s_1, \ldots, x^s_k) \in R^S$$

$$S \models \exists x \phi \iff S[x := u] \models \phi \text{ for some } u \in U^s$$

$$S \models \forall x \phi \iff S[x := u] \models \phi \text{ for every } u \in U^s$$

What is the link to QBF:
When you define the Universe U^S as $U^S = \{true, false\}$ FO - First order logic becomes (almost) QBF

Example for a relation: Database:

movies	year
Bond	2020
Harry Potter	2010
...	...

(Bond,2020) $\in R$
If we want to use such a tuple we write in a formula $R(x_1, x_2)$
while x_1 stands for the movie name and x_2 stands for the release year.

2.1.1 Example 1:

If all humans are mortal , and
Socrates is human ,
then Socrates is mortal .

$\phi(y_0) = ((\forall x A(x) \rightarrow B(x)) \wedge A(y_0)) \rightarrow B(y_0)$

Lets define what is the Universe and what the relation-symbols are
Socrates, Plato -> humans
Cyclop -> mythological figure that can also die (is killed by Ulisses in the Illias)
Jupyter -> god so not mortal

$$S : \begin{cases} U^S & = \{\text{Socrates, Plato, Cyclop, Jupyter}\} \\ A^S & = \{\text{Socrates, Plato}\} \text{ -> humans} \\ B^S & = \{\text{Socrates, Plato, Cyclop}\} \text{ -> Mortals} \\ y_0^S & = \text{Socrates} \end{cases}$$

We look at unary relations. That means that all relations are subsets of the universe U^S. The universe can not be empty.

2.1.2 Example 2:

There is a node in the graph
that is isolated
from all other nodes

$$\phi() = \exists x \forall y (\neg(x = y) \implies \neg E(x, y))$$

relationship symbols = {=,E}
free variables = {} -> all variables are part of Quantifiers
The 2 relations are binary relations

$$S : \begin{cases} U^S & = \{\text{Nodes of a graph}\} \\ E^s & = \{\text{edges of a graph}\} \\ =^S & = \{(x, x) | x \in U^S\} \text{ -> quality relation} \end{cases}$$

1. $\neg(x = y)$ means y is different from x
2. $E(x, y)$ returns true if they have a edges that connect themselv

2.1.3 Example 3:

There is a man such That
when he runs
everybody runs

$$\phi() = \exists x (R(x) \implies \forall y R(y))$$

2 variables -> none of them free
1 relation

$$S: \begin{cases} U^S & = \{\text{Ben,Han,Leia,Luke}\} \\ R^s & = \{\text{Ben, Han}\} \end{cases}$$

Does the structure S hold?

Yes because we have a $\exists$. It has to be only true for one of the elements of the universe:

- As Leia and Luke are not running it holds true. As they do not run they do not make everybody else run. i.e. the make everybody else run is not triggered i.e. the formula is true for Leia and Luke.
- If we choose Ben and Han the formula is false as Ben is running and while he runs only Han runs and vice-versa. Because Leia and Luke are not running the formula is false.
- If we exchange the $\exists x$ by $\forall x$ everybody needed to run when one of the elements run i.e. the formula could not be True as Leia and Luke never run.

What happens when we change the Structure:

$$S': \begin{cases} U^S & = \{\text{Ben,Han,Leia,Luke}\} \\ R^s & = \{\text{Ben, Han,Leia,Luke}\} \end{cases}$$

Then the formula is true for both:

$$\phi() = \exists x(R(x) \implies \forall y R(y))$$

and

$$\phi() = \forall x(R(x) \implies \forall y R(y))$$

2.1.4 Example 4:

R is a function

$\phi = \forall x \exists y R(x,y) \wedge (\forall z R(x,z) \implies z = y)$

1. $\forall x \exists y R(x,y)$ means that the function is not a partial function i.e. for every $x \in X$ there is a $y \in Y$ that can be brought into relation.\
2. $(\forall z R(x,z) \implies z = y)$ means that there is a 1 to 1 relationship between the elements of X and Y. It is not possible two relate two different objects y and z to the same x
   i.e.
   is allowed

- R(x,b) -> k
- R(x,n) -> m
  is not allowed
- R(x,b) -> k
- R(x,n) -> k

This formula means that no tuples of elements from the source set X should not point to the same element in the goal set and that all elements of X have a element with which it can be brought into relation.

If one writes:
$R(x) = \ldots$ it means $R(x) = \forall x \exists y R(x,y) \wedge (\forall z R(x,z) \implies z = y)$

2.1.5 Example 5:

"$+$ is commutative"

$$\phi = \forall x \forall y (x + y = y + x)$$

Learnings x is a three way relation (ternary operation) two inputs x,y point to one output z. One could write this relation also differently

$$\forall x \forall y \forall z \forall z' (+(x, y, z) \wedge +(x, y, z')) \implies z = z'$$

2.1.6 Example 6:

"+ admits 0 and inverses"

$$\phi = \exists x \forall y \exists z (x + y = y \wedge y + z = x)$$

- $\exists x \forall y x + y = y$ means there are 0 (neutral elements)
- $\exists x \forall y \exists z y + z = x$ means there are inverses

2.1.7 Example 7:

'f is continuous'

$$\phi = \forall \epsilon \forall x \exists \delta \forall y (||x - y|| < \delta \implies ||f(x) - f(y)|| < \epsilon)$$

No free variables!

$$S : \begin{cases} U^S & = \mathbb{R} \\ ||\ldots||_2 & \subseteq U^S \times U^S \\ +/- & \subseteq U^S \times U^S \times U^S \\ < & \subseteq\subseteq U^S \times U^S \\ f & \subseteq U^S \times U^S \end{cases}$$

'f is uniformly continuous'

$$\phi = \forall \epsilon \exists \delta \forall x \forall y (||x - y|| < \delta \implies ||f(x) - f(y)|| < \epsilon)$$

No free variables

$$S : \begin{cases} U^S & = \mathbb{R} \\ ||\ldots||_2 & \subseteq U^S \times U^S \\ +/- & \subseteq U^S \times U^S \times U^S \\ < & \subseteq\subseteq U^S \times U^S \\ f & \subseteq U^S \times U^S \end{cases}$$

Note

uniform continuous $\implies$ continuous
Example: \frac{1}{x} is continuous but not uniformly continuous in $\mathbb{R}$

Here we got some exercises Exercises Verification 7. March class 4