We can do even better than that! Looking again at the definition of the $\mathcal{A}, t \models \varphi$ relation, we note that we can reformulate it in such a way, that truth only depends on the truth of proper subformulae at the same time, or a (non-proper) subformula at the next point in time.

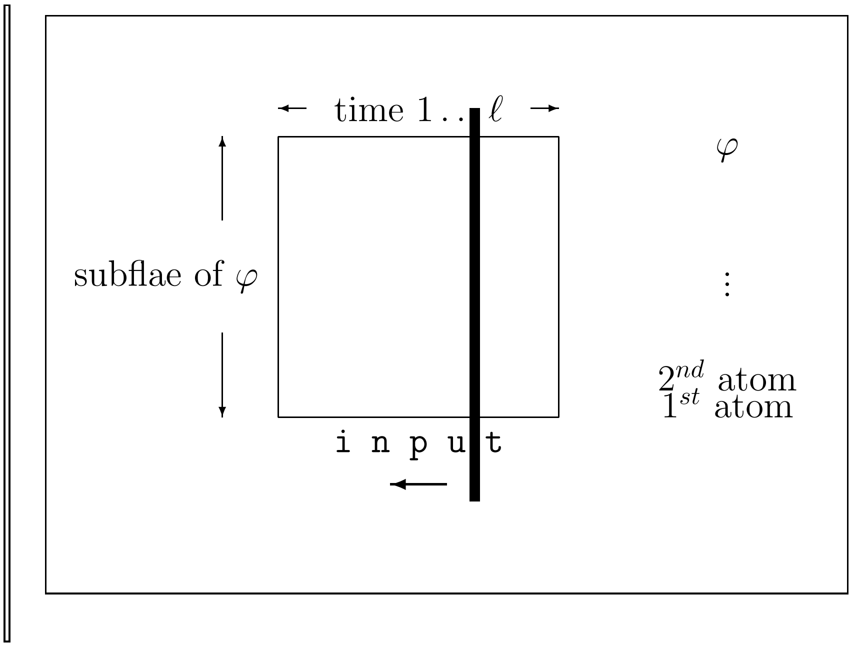
LTL Model Checking

t		t		$t{+}1$
$\mathcal{A}, t \models \mathtt{X}\varphi$	iff			$\mathcal{A}, t{+}1 \models \varphi$
$\mathcal{A}, t \models \mathtt{F}\varphi$	iff	$\mathcal{A}, t \models \varphi$	or	$\mathcal{A}, t{+}1 \models \mathtt{F}\varphi$
$\mathcal{A}, t \models \mathtt{G}\varphi$	iff	$\mathcal{A}, t \models \varphi$	and	$\mathcal{A}, t{+}1 \models \mathtt{G}\varphi$
				in case $t + 1 \leq \ell$
$\mathcal{A}, t \models \varphi\mathtt{U}\psi$	iff	$\mathcal{A}, t \models \psi$	or	
		$\mathcal{A}, t \models \varphi$	and	$\mathcal{A}, t{+}1 \models \varphi\mathtt{U}\psi$

On the one hand, the above proof now gives the better time complexity, since the amount of lookup for any particular filed is now constant.

> Time $\mathcal{O}(|\mathcal{A}| \cdot |\varphi|)$

Moreover, the amount of information we have to store only depends on the size of the formula. We can just sweep through the word/trace and update the information about all subformulae.



Let's see this form of model checking in an example.

Example for LTL Model Checking

Formula $\mathtt{G}(y \to y\mathtt{U}r)$. *That this* $\mathtt{G}(\neg y \vee y\mathtt{U}r)$.

$\mathcal{A}$									
r	1	1	1	0	0	0	0	0	1
y	0	0	1	0	0	0	1	1	0
g	0	0	0	1	1	1	0	0	0
$\mathtt{G}(y \to y\mathtt{U}r)$									
$y \to y\mathtt{U}r$									
$y\mathtt{U}r$									
$\neg y$									

Note that, when working our way from the end of the trace to its beginning, we could, at any point, forget any future columns after the earliest one we filled.

This is precisely the characteristic behaviour of an automaton walking (backwards) over the word.

LTL and Automata

For φ construct DFA $\mathcal{M}_\varphi$ with $L(\mathcal{M}_\varphi) = \{w | w^{-1} \models \varphi\}$.

States: sets of sub-formulae of φ.
indicating with formulae hold at a given position

Transitions: Given

- previous state $\{\psi \mid \mathcal{A}, t{+}1 \models \psi\}$
- t'th letter of w, i.e., local properties of $\mathcal{A}$ at time t

determine $\{\psi \mid \mathcal{A}, t \models \psi\}$ by the rules seen.

This description by an automata has several important consequence. First of all, we get decidability of the satisfiability problem. This was not a priory clear. As opposed to propositional logic, we didn't have an a priory bound on the candidate structures (as words can have arbitrary length!).

> **Theorem.** Satisfiability for an LTL-formula φ over finite words can be decided in *space* polynomial $|\varphi|$.
>
> **Proof.** Decide the emptiness-problem for the automaton associated with φ.
> *(Since a state is described by $|\varphi|$ many bits, we are in non-deterministic space $\mathcal{O}(|\varphi|)$, which is in deterministic space $\mathcal{O}(|\varphi|^2)$.)*

Moreover, the fact that the number of states of the automaton has "only" $2^{|\varphi|}$ states gives us a bound on the size of the smallest structure satisfying an LTL property.

Corollary. If an LTL-formula φ can be satisfied by a finite structure, then it can also be satisfied by a structure of size at most $2^{|\varphi|}$.

Also note that from a DFA for w^{-1} we can get directly an NFA for w (and therefore also an DFA at the price of an exponential increase in the number of states).

All LTL-definable properties are regular.

In particular, we have found a bound on the expressibility of LTL. Just take your favourite non-regular language and LTL cannot express it.

There is no LTL-formula expressing "the word looks like $\underbrace{\mathtt{aa}\ldots\mathtt{a}}_{n}\underbrace{\mathtt{bb}\ldots\mathtt{b}}_{n}$"

As it turns out, this bound is not sharp. There are regular languages that *cannot* be expressed using an LTL formula. However, proving this fact is beyond the scope of this course.

2.3 First-Order Logic

As mentioned, the structures we have been looking at are first-order structures. The language usually used to describe them is first-order logic. We'll see that first-order logic can, without a significant overhead, express everything we could express in linear temporal logic. However, the complexity of model checking and the satisfiability problem will turn out to be much harder for first-order logic.

Recall the general setting of the structures we're talking about.

First-Order Structures

Definition. A first-order structure $\mathcal{A} = (A, R_1, R_2, \ldots)$ is given by

- A set A, called the "universe" of the structure
- Relations $R_1, R_2, \ldots$ on A.
  I.e., each $R_i \subseteq A^{n_i}$ for some n_i, called the arity.

So far, the elements of the universe were always handled implicitly. So, we didn't have to have names for them. Now we assume some fixed infinite set $V = \{x_1, x_2, \ldots\}$ of first order variables. Our notion of (first-order) formula will implicitly refer to this set.

We will use $x, y, z, \ldots$ to denote variables.

Syntax of First-Order Logic

The set $\mathsf{FOL}[R_1, \ldots, R_n]$ of first-order formulae over the relation symbols $R_1, \ldots, R_n$ is freely generated as follows.

- $(x = y)$ for variables x, y
- $R_j x_{i_1} \ldots x_{i_{n_j}}$ if R_j is of arity n_j
- $\top$, $\bot$, $\neg\phi$, $(\phi \wedge \psi)$, $(\phi \vee \psi)$ for formulae ϕ, ψ
- $\forall x\varphi$ and $\exists x\varphi$ for ϕ a formula and x a variable

Intuitively, we consider the variable x to be bound in $\forall x\varphi$. This intuition is reflected in our definition of the set of "free variables" of a formula.

$$
\begin{array}{lcl}
\mathsf{fv}(x = y) & = & \{x, y\} \\
\mathsf{fv}(Ry_1 \ldots y_k) & = & \{y_1, \ldots, y_k\} \\
\mathsf{fv}(\top) = \mathsf{fv}(\bot) & = & \emptyset \\
\mathsf{fv}(\neg\varphi) & = & \mathsf{fv}(\varphi) \\
\mathsf{fv}((\varphi \wedge \psi)) = \mathsf{fv}((\varphi \vee \psi)) & = & \mathsf{fv}(\varphi) \cup \mathsf{fv}(\psi) \\
\mathsf{fv}(\forall x\varphi) = \mathsf{fv}(\exists x\varphi) & = & \mathsf{fv}(\varphi) \setminus \{x\}
\end{array}
$$