Definition 1. Let $\Gamma \in \mathrm{Cont}$, $M \in \mathrm{Exp}$, $A \in \mathrm{Type}$, $\Gamma \vdash M : A$ is defined by recursivelly on the height of M:

$$\frac{}{\Gamma \cup \{x : A\} \vdash x : A}$$

$$\frac{\Gamma \vdash P : A \to B \qquad \Gamma \vdash Q : A}{\Gamma \vdash PQ : B}, \qquad \frac{\Gamma \cup \{x : A\} \vdash P : B}{\Gamma \vdash \lambda x.P : A \to B}$$

Proposition 1. There is an alternating polynomial time algorithm, and thus also a deterministic polynomial space algorithm to determine whether a given type A is inhabited in a given basis Γ in simply-typed lambda calculus.

Proof. According to the Normal Form Theorem of lambda calculus, it is enough to produce a term of the long normal form:

$$\lambda x_1 \dots \lambda x_n . y M_1 \dots M_m$$

where y is a variable and $M_1 \dots M_m$ are in normal forms, Urzyczyn (1995).

(Without this, we only claim that among the long normal forms there the inhabitation is decidable in APTIME.)

The following procedure returns $\mathrm{inhab}_o(\Gamma \vdash ? : A)$ such that $\Gamma \vdash \mathrm{inhab}_o(\Gamma \vdash ? : A) : A$ as an output, if it holds, and produces an answer "reject" otherwise.

Procedure $\mathrm{inhab}(\Gamma \vdash ? : A)$

if A atomic **then**

  choose non-deterministically $y :\ (y : A_1 \to \dots \to A_m \to A) \in \Gamma$

  if $\exists y :\ (y : A_1 \to \dots \to A_m \to A) \in \Gamma$ **then**

    if $\forall i = 1 \dots m :$ $\mathrm{inhab}(\Gamma \vdash ? : A_i)$ is (**parallel**) **accepted then**

      $\mathrm{inhab}(\Gamma \vdash ? : A)$ is **accepted**

      let $\mathrm{inhab}_o(\Gamma \vdash ? : A) = y\, \mathrm{inhab}_o(\Gamma \vdash ? : A_1) \dots \mathrm{inhab}_o(\Gamma \vdash ? : A_m)$

    else $\mathrm{inhab}(\Gamma \vdash ? : A)$ is **rejected**

    end if

  else $\mathrm{inhab}(\Gamma \vdash ? : A)$ is **rejected**

  end if

else if $A = A_1 \to A_2$ **then**

choose non-deterministically $x: \ (x : A_1) \in \Gamma$

if $\exists x: \ (x : A_1) \in \Gamma$ **then**

let $\Gamma' = \Gamma$

else let $\Gamma' = \Gamma \cup \{x : A_1\}$ with fresh x

end if

if inhab($\Gamma' \vdash ? : A_2$) is **accepted then**

inhab($\Gamma \vdash ? : A$) is **accepted**

let inhab$_o$($\Gamma \vdash ? : A$) $= \lambda x.$inhab$_o$($\Gamma' \vdash ? : A_2$)

else if inhab($\Gamma' \vdash ? : A_2$) is **rejected then**

inhab($\Gamma \vdash ? : A$) is **rejected**

end if

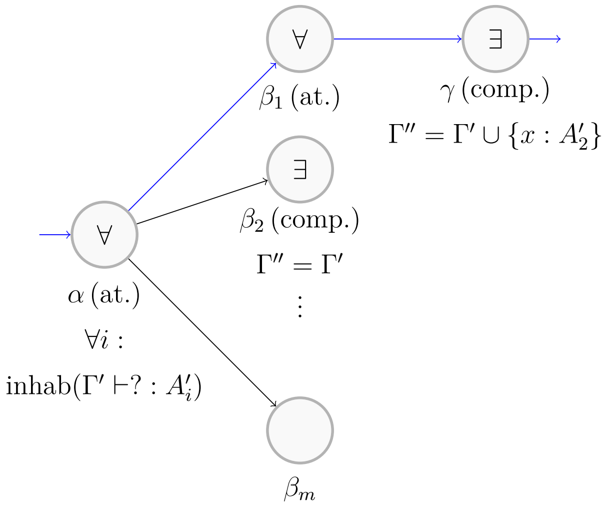
end if

Informally,

(1) If A is a type variable, or in other words an atomic type (the one that does not contain an arrow), then M is an application of a variable to a sequence of terms (basically functional application defined above). We then non-deterministically choose a variable, declared in Γ to be of type $A_1 \to \cdots \to A_m \to A$. If there is no such variable, we reject. If $m > 0$, we answer in parallel the questions if A_i are inhabited in Γ.

(2) If a certain type is in the form: $A_1 \to A_2$ then the lambda expression must be an abstraction $M \equiv \lambda x.M'$, therefore we search for an M' that satysfies $\Gamma, x : A_1 \vdash M' : A_2$. It means that in such cases what we are supposed to do is incorporate a variable with type A_1 into our base and given we do that a new lambda expression M' with type A_2 becomes derivable. The key is that upon finding out the characteristics of a certain type, the algorithm decides whether the preconditions and the route for that type are satysfied. The reason why it is a polynomial time procedure is that the number of steps the algorithm takes only grows by at most a quadratic rate as a function of context size.

The size of the input is the number of sub-formulas in $\Gamma \cup \{A\}$, say n. We have to give an upper bound for the length of the longest computational path. Suppose, α is a universal state in the case "A is atomic" and the question is whether inhab($\Gamma' \vdash ? : A_i'$) is accepted for all $i = 1...m$ or not. If β is also an "A is atomic" case (β_1) or an existential state in the compound "$A = A_1 \to A_2$" case (β_2), then Γ remains unchanged and the number of such transitions (with unchanged Γ') is not greater than the number of questions of the form $\Gamma' \vdash ? : B$ where Γ' is fixed, i.e. the number of all subformulas in the original $\Gamma \cup \{A\}$ which is n. If γ is a state in the compound "$A = A_1 \to A_2$" case and $\Gamma'' = \Gamma' \cup \{x : A_2\}$ is a proper expansion, then *along one computational path,* the number of such expansions is not greater than the number of all subformulas in the original $\Gamma \cup \{A\}$, i.e. n. Hence, the depth of recursion is at most the number of possible proper expansions times the number of questions in a given expansion, i.e. n^2.



$\square$

The above PTIME *alternating* algorithm employs two magicians. One works in case (1), who gives us the lucky path leading us to the answer "accept", and the other one in case (2), leading us to the answer "reject". This is crucial because otherwise the number of steps during the solution is much larger, hence the distinction in the proposition, "alternating polynomial time and deterministic polynomial space" algorithms.

Proposition 2. There is an alternating linear time algorithm deciding whether the relation

$$\Gamma \vdash N : A$$

holds or not, if context Γ, *normal* expression N, and type A are given.

Proof. Realization of type checking of a normal expression:

Input: Γ, N, A, where N is normal.

Input size: the height $|N|$ of the syntax tree of N.

Output: the Boolean value **accept/reject**.

Procedure typecheck$_a$ accepts the input (Γ, N, A) iff $\Gamma \vdash N : A$ holds.

procedure typecheck$_a$

```
if  N = λx.N′ then
    if  A = A₁ → A₂ then
        do typecheck_a(Γ ∪ {x : A₁}, N′, A₂)
        if  typecheck_a accepts (Γ ∪ {x : A₁}, N′, A₂)  then
            accept  (Γ, N, A)
        else reject  (Γ, N, A)
        end if
    else reject  (Γ, N, A)
    end if
else if  N = N′N″  then
    choose non-deterministically A₃ ∈ Sub(Γ ∪ {A}) such that
        { typecheck_a accepts (Γ, N′, A₃ → A)  and
        { typecheck_a accepts (Γ, N″, A₃)
    if  it is successful  then  accept (Γ, N, A)
    else reject (Γ, N, A)
    end if
else if  N = x  then
    choose non-deterministically x such that (x : A) ∈ Γ
    if  it is successful  then  accept (Γ, N, A)
    else  reject (Γ, N, A)
    end if
end if
```

The depth of recursion is preciselly $|N|$, hence the alternating algorithm's runtime is $\boxed{\mathcal{O}(|N|)}$

□

Conjecture 1. There is a deterministic fixed parameter linear algorithm deciding whether the relation

$$\Gamma \vdash N : A$$

holds or not, if context Γ, *normal* expression N, and type A are given.

Realization of type checking of a normal expression:

Input: Γ, N, A, where N is normal.

Parameter: $k = |\mathrm{Sub}(\Gamma \cup \{A\})|$ (the number of subformulas in $\Gamma \cup \{A\}$)

Input size: the height $|N|$ of the syntax tree of N.

Output: the Boolean value **accept**/**reject**.

Procedure $\mathsf{typecheck}_p$ accepts the input (Γ, N, A) iff $\Gamma \vdash N : A$ holds.

The runtime of $\mathsf{typecheck}_p$ is less than $\boxed{2k^2 \cdot |N|}$.

Conjecture 2. There is a PTIME algorithm deciding whether the relation

$$\Gamma \vdash N : A$$

holds or not, if context Γ, *normal* expression N, and type A are given.

Realization of type checking of a normal expression:

Input: Γ, N, A, where N is normal.

Input size: $n = \max\{|\mathrm{Sub}(\Gamma \cup \{A\})|, |N|\}$

Output: the Boolean value **accept/reject**.

Procedure $\mathsf{typecheck}_d$ accepts the input (Γ, N, A) iff $\Gamma \vdash N : A$ holds.

The runtime of $\mathsf{typecheck}_d$ is less than $\boxed{2|N|^3}$.