FIG. 3.2 – Algorithme descendant le plus (trop !) naïf

```
// la fonction principale est tdlrp : top-down left-right parsing
// α est la protophrase courante, u l'entrée à reconnaître
tdlrp(α, u)
{
    if (α = u) : return true
    α = u_1 u_2 ... u_k A γ
    while (∃ A → β) do
        if tdlrp(u_1 u_2 ... u_k β γ, u) : return true
    return false
}
```

FIG. 3.3 – Un exemple de dérivation descendante, avec comparaison u/α

Exemple $S \to aSbS \mid bSaS \mid \varepsilon$, mot : $u = abba$
Choix : d'abord le non-terminal le plus à gauche (*leftmost*). Règles considérées de gauche à droite.
Première règle :
$S \longrightarrow a\underline{S}bS \longrightarrow a\boxed{aSbS}bS$!!! mot dérivé : $aa\ldots \neq$ mot cherché : $ab\ldots$
Deuxième règle :
$S \longrightarrow a\underline{S}bS \longrightarrow a\boxed{b\underline{S}aS}bS$!!! mot dérivé : au moins 4 lettres, qui ne sont pas les bonnes
Troisième règle + première règle :
$S \longrightarrow a\underline{S}bS \longrightarrow a\boxed{\varepsilon}b\underline{S} \longrightarrow ab\boxed{aSbS}$!!! mot dérivé : $aba\ldots \neq$ mot cherché : $abb\ldots$
Troisième règle + deuxième règle :
$S \longrightarrow a\underline{S}bS \longrightarrow a\boxed{\varepsilon}b\underline{S} \longrightarrow ab\boxed{bSaS}$ ... ...
Troisième règle + deuxième règle + troisième règle + troisième règle :
$S \longrightarrow a\underline{S}bS \longrightarrow a\boxed{\varepsilon}b\underline{S} \longrightarrow ab\boxed{b\underline{S}aS} \longrightarrow abb\boxed{\varepsilon}a\underline{S} \longrightarrow abba\boxed{\varepsilon}$

FIG. 3.4 – Algorithme descendant standard, version 1

```
// la fonction principale est tdlrp : top-down left-right parsing
// α est la protophrase courante, u l'entrée à reconnaître
tdlrp(α, u)
{
    if (α = u) : return true
    α = u_1 u_2 ... u_k A γ
    while (∃ A → β) do
        où β = u_{k+1} ... u_{k+l} δ avec δ = ε ou δ = Bη, et l ∈ ℕ
        if tdlrp(u_1 u_2 ... u_k β γ, u) : return true
    return false
}
```