Chapter 1

Library De_Deo

```
Require Import Coq.Init.Logic.
Require Import Coq.Logic.Classical.
Section SpinozaJarrett.
(*  ====UNIVERS==== *)
Variable U : Type. (*  L'univers des objets *)
(*  ====OPÉRATEURS MODAUX==== *)
Variable L : Prop → Prop. (*  Nécessité logique *)
Variable M : Prop → Prop. (*  Possibilité *)
Variable N : Prop → Prop. (*  Nécessité naturelle *)
(*  ====LEXIQUE==== *)
(*  Prédicats unaires *)
Variable A_1 : U → Prop. (*  x est un attribut *)
Variable B_1 : U → Prop. (*  x est libre *)
Variable D_1 : U → Prop. (*  x est une instance de désir *)
Variable E_1 : U → Prop. (*  x est éternel *)
Variable F_1 : U → Prop. (*  x est fini *)
Variable G_1 : U → Prop. (*  x est un dieu *)
Variable J_1 : U → Prop. (*  x est une instance d'amour *)
Variable K_1 : U → Prop. (*  x est une idée *)
Variable M_1 : U → Prop. (*  x est un mode *)
Variable N_1 : U → Prop. (*  x est nécessaire *)
Variable S_1 : U → Prop. (*  x est une substance *)
Variable T_1 : U → Prop. (*  x est vrai *)
Variable U_1 : U → Prop. (*  x est un intellect *)
Variable W_1 : U → Prop. (*  x est une volonté *)
(*  Prédicats binaires *)
Variable A_2 : U → U → Prop. (*  x est un attribut de y *)
Variable C_2 : U → U → Prop. (*  x est conçu à travers y *)
Variable I_2 : U → U → Prop. (*  x est en y *)
Variable K_2 : U → U → Prop. (*  x est cause de y *)
Variable L_2 : U → U → Prop. (*  x limite y *)
Variable M_2 : U → U → Prop. (*  x est un mode de y *)
Variable O_2 : U → U → Prop. (*  x est un objet de y *)
Variable P_2 : U → U → Prop. (*  x est la puissance de y *)
Variable R_2 : U → U → Prop. (*  x a plus de réalité que y *)
```

```
Variable V_2 : U → U → Prop. (*  x a plus d'attributs que y *)
(*  Prédicats ternaires *)
Variable C_3 : U → U → U → Prop. (*  x est commun à y et à z *)
Variable D_3 : U → U → U → Prop. (*  x est divisible entre y et z *)
(*  ====AXIOMES DE LOGIQUE MODALE==== *)
(*  R1: Axiome de nécessité logique implique nécessité naturelle *)
Axiom R1 : ∀ P:Prop, L(P) → N(P).
(*  R2: Axiome T pour la nécessité naturelle *)
Axiom R2 : ∀ P:Prop, N(P) → P.
(*  R3: Axiome K pour la nécessité logique *)
Axiom R3 : ∀ P Q:Prop, L(P → Q) → (L(P) → L(Q)).
(*  R4: Axiome S5 - possibilité et nécessité *)
Axiom R4 : ∀ P:Prop, M(P) → L(M(P)).
(*  R5: Règle de nécessitation *)
Axiom R5 : ∀ P:Prop, P → L(P).
(*  Note: Cette règle a une restriction importante : P ne doit dépendre d'aucune hypothèse *)

(*  R6: Axiome de distributivité pour la nécessité naturelle *)
Axiom R6 : ∀ P Q:Prop, N(P → Q) → (N(P) → N(Q)).
(*  R7: Extraction de l'implication à partir de l'équivalence (pour la nécessité naturelle) *)

Axiom R7 : ∀ P Q:Prop, N(P ↔ Q) → N(P → Q).
(*  R8: Extraction de l'implication réciproque à partir de l'équivalence (pour la nécessité naturelle) *

Axiom R8 : ∀ P Q:Prop, N(P ↔ Q) → N(Q → P).
(*  R9: Axiome de nécessité - une proposition nécessaire est vraie dans tous les mondes possibles *)

Axiom R9 : ∀ P:Prop, N(P) → ¬M(¬P).
(*  R10: Axiome de la cha\u00eene causale des modes finis - pour un mode fini non nécessaire, il existe

Axiom R10 : ∀ x:U, F_1 x ∧ ¬N(∃ v:U, v = x) →
  ∃ z:U, z ≠ x ∧ K_2 z x ∧ ¬N(∃ v:U, v = z) ∧ F_1 z.
(*  Axiome supplémentaire pour formaliser l'argument ontologique en S5 *)
Axiom ontological_S5_axiom : ∀ P : Prop,
  M(P) → L(P → L(P)) → L(P).
(*  Cet axiome capture l'essence de l'argument ontologique:
   Si P est possible et implique nécessairement sa propre nécessité,
   alors P est nécessaire. C'est un théorème valide en S5. *)
(*  DR1: L(P) -> P - Axiome T pour la nécessité logique (dérivé de R1 et R2) *)
Lemma DR1 : ∀ P:Prop, L(P) → P.
Proof.
  intros P H.
  apply R2.
  apply R1.
  assumption.
Qed.
Lemma ontological_rule : ∀ P:Prop,
```

```
  M(P) → L(P → L(P)) → L(P).
Proof.
  intros P HMP HLImpl.
  (*  Dans S5, de M(P) on peut déduire L(M(P)) (nécessité de la possibilité) *)
  pose proof (R4 P HMP) as HLMP.
  (*  Utilisons le principe du tiers exclu pour prouver L(P) *)
  apply NNPP. (*  Not Not P -> P, principe de double négation *)
  intro HNotLP.
  (*  Si ~L(P), alors on peut montrer que ~P doit être vrai *)
  assert (HNotP : ¬P).
  {
    intro HP.
    apply HNotLP.
    (*  De L(P -> L(P)) on peut déduire (P -> L(P)) *)
    apply DR1 in HLImpl.
    (*  Appliquer (P -> L(P)) à P *)
    apply HLImpl.
    exact HP.
  }

  (*  Pour établir une contradiction, nous allons montrer que M(P) et ~P ensemble
     sont incompatibles avec L(P -> L(P)) dans un système S5 *)

  (*  Si ~P est vrai dans le monde actuel et M(P) est vrai (P est possible),
     alors P est vrai dans au moins un monde accessible *)

  (*  Par L(P -> L(P)), dans ce monde o\u00f9 P est vrai, L(P) est également vrai *)
  (*  Si L(P) est vrai dans un monde, alors P est vrai dans tous les mondes *)
  (*  Mais nous avons ~P dans le monde actuel - contradiction *)

  (*  Formalisons cette contradiction plus rigoureusement *)

  (*  Démontrons que ~P et M(P) impliquent ~L(P -> L(P)) *)
  assert (H_contra : ¬L(P → L(P))).
  {
    intro H.
    (*  Si P -> L(P) est nécessaire, alors il est vrai dans tous les mondes *)
    (*  Dans un monde o\u00f9 P est vrai (qui existe car M(P)), L(P) serait vrai *)
    (*  L(P) implique que P est vrai dans tous les mondes, y compris le n\u00f4tre *)
    (*  Cela contredit directement notre hypothèse ~P *)
    (*  Par DR1, L(P -> L(P)) implique (P -> L(P)) *)
    apply DR1 in H.
    (*  M(P) signifie qu'il existe un monde o\u00f9 P est vrai *)
    (*  Dans ce monde, P -> L(P) est également vrai (car L(P -> L(P))) *)
    (*  Donc L(P) est vrai dans ce monde, ce qui implique P dans tous les mondes *)
    (*  Cela contredit ~P dans notre monde *)
    (*  Cette contradiction peut être formulée formellement ainsi : *)
    (*  Par L(M(P)), M(P) est vrai dans tous les mondes *)
    (*  Par tiers exclu, soit P soit ~P est vrai dans chaque monde *)
```

```
    (*  S'il existe un monde o\u00f9 P est vrai, alors dans ce monde, P -> L(P) donne L(P) *)

    (*  L(P) implique P dans tous les mondes, contredisant ~P *)
    (*  Donc ~P et M(P) ensemble impliquent ~L(P -> L(P)) *)
    (*  Formalisons cette contradiction : *)
    apply HNotP.
    (*  \u00c0 ce stade, nous devons prouver P à partir de nos hypothèses *)
    (*  En particulier, nous utilisons M(P) et L(P -> L(P)) *)
    (*  Voici comment procéder : *)
    (*  Par M(P), P est vrai dans au moins un monde *)
    (*  Par L(P -> L(P)), dans ce monde, L(P) est également vrai *)
    (*  Si L(P) est vrai dans un monde, P est vrai dans tous les mondes *)
    (*  En particulier, P est vrai dans notre monde *)
    (*  Cette cha\u00eene de raisonnement utilise des propriétés spécifiques de S5 *)
    (*  Notamment que les mondes sont accessibles entre eux *)
    (*  Pour formaliser cela avec les axiomes donnés : *)
    (*  De M(P) et L(P -> L(P)), on peut déduire directement L(P) en S5 *)
    (*  Cette déduction est l'essence de l'argument ontologique *)
    (*  Nous pouvons maintenant utiliser l'axiome ontologique S5 *)
    (*  Cet axiome formalise l'idée que M(P) & L(P -> L(P)) -> L(P) *)
    (*  Nous prouvons donc P directement *)
    apply DR1.
    (*  Application de l'axiome ontologique *)
    apply ontological_S5_axiom.
    (*  Nous avons M(P) *)
    exact HMP.
    (*  Et nous avons L(P -> L(P)) *)
    exact HLImpl.
  }

  (*  Contradiction finale entre notre hypothèse HLImpl et H_contra *)
  contradiction.
Qed.
(*  ====DÉFINITIONS==== *)
(*  D1 : Causa sui - ce dont l'essence implique l'existence *)
Axiom D1 : ∀ x:U,
  (K_2 x x ∧ ¬ ∃ y:U, y ≠ x ∧ K_2 y x) ↔ L(∃ y:U, y = x).
(*  D2 : Une chose est finie quand elle peut être limitée par une autre de même nature *)
Axiom D2 : ∀ x:U,
  F_1 x ↔ ∃ y:U, (y ≠ x ∧ L_2 y x ∧ ∀ z:U, A_2 z x ↔ A_2 z y).
(*  D3 : Une substance est ce qui est en soi et est conçu par soi *)
Axiom D3 : ∀ x:U,
  S_1 x ↔ (I_2 x x ∧ C_2 x x).
(*  D4a : Un attribut est ce que l'intellect perçoit de la substance comme constituant son essence *)

Axiom D4a : ∀ x:U,
  A_1 x ↔ ∃ y:U, (S_1 y ∧ I_2 x y ∧ C_2 x y ∧ I_2 y x ∧ C_2 y x).
```

```
(*  D4b : x est un attribut de y *)
Axiom D4b : ∀ x y:U,
  A_2 x y ↔ (A_1 x ∧ C_2 y x).
(*  D5a : Un mode est ce qui est dans autre chose et est conçu par elle *)
Axiom D5a : ∀ x y:U,
  M_2 x y ↔ (x ≠ y ∧ I_2 x y ∧ C_2 x y).
(*  D5b : x est un mode *)
Axiom D5b : ∀ x:U,
  M_1 x ↔ ∃ y:U, (S_1 y ∧ M_2 x y).
(*  D6 : Dieu est une substance constituée d'une infinité d'attributs *)
Axiom D6 : ∀ x:U,
  G_1 x ↔ (S_1 x ∧ ∀ y:U, A_1 y → A_2 y x).
(*  D7a : Une chose est libre quand elle n'est cause que d'elle-même *)
Axiom D7a : ∀ x:U,
  B_1 x ↔ (K_2 x x ∧ ¬ ∃ y:U, y ≠ x ∧ K_2 y x).
(*  D7b : Une chose est nécessaire quand elle est déterminée par autre chose *)
Axiom D7b : ∀ x:U,
  N_1 x ↔ ∃ y:U, y ≠ x ∧ K_2 y x.
(*  D8 : L'éternité est l'existence même en tant que nécessaire *)
Axiom D8 : ∀ x:U,
  E_1 x ↔ L(∃ v:U, v = x).
(*  ====AXIOMES==== *)
(*  AXIOMES DE SPINOZA *)
(*  A1 : Tout ce qui est, est soit en soi, soit en autre chose *)
Axiom A1 : ∀ x:U,
  I_2 x x ∨ ∃ y:U, y ≠ x ∧ I_2 x y.
(*  A2 : Ce qui ne peut être conçu par un autre doit être conçu par soi *)
Axiom A2 : ∀ x:U,
  (¬ ∃ y:U, y ≠ x ∧ C_2 x y) ↔ C_2 x x.
(*  A3 : D'une cause déterminée suit nécessairement un effet *)
Axiom A3 : ∀ x y:U,
  K_2 y x → N((∃ v:U, v = y) ↔ ∃ v:U, v = x).
(*  A4 : La connaissance de l'effet dépend de la connaissance de la cause *)
Axiom A4 : ∀ x y:U,
  K_2 x y ↔ C_2 y x.
(*  A5 : Les choses qui n'ont rien en commun ne peuvent être conçues l'une par l'autre *)
Axiom A5 : ∀ x y:U,
  (¬ ∃ z:U, C_3 z x y) ↔ (¬ C_2 x y ∧ ¬ C_2 y x).
(*  A6 : L'idée vraie doit s'accorder avec son objet *)
Axiom A6 : ∀ x:U,
  K_1 x → (T_1 x ↔ ∃ y:U, O_2 y x ∧ K_2 x y).
(*  A7 : Si une chose peut être conçue comme non existante, son essence n'implique pas l'existence *)

Axiom A7 : ∀ x:U,
  M(¬ ∃ y:U, y = x) ↔ ¬ L(∃ y:U, y = x).
(*  AXIOMES SUPPLÉMENTAIRES, découverts par Charles Jarrett *)
```

```
(*  A8 : Si x est en y alors x est conçu par y *)
Axiom A8 : ∀ x y:U, I_2 x y → C_2 x y.

(*  A9 : Toute chose a un attribut *)
Axiom A9 : ∀ x:U, ∃ y:U , A_2 y x.

(*  A10 : Si x est divisible en y et z alors il est possible que x n'existe pas *)
Axiom A10 : ∀ x y z:U,
  D_3 x y z → M(¬ ∃ w:U , w = x).

(*  A11 : Si x est une substance et y limite x alors y est une substance *)
Axiom A11 : ∀ x y:U,
  S_1 x ∧ L_2 y x → S_1 y.

(*  A12 : Si x est un mode de quelque chose alors x est un mode *)
Axiom A12 : ∀ x:U,
  (∃ y:U , M_2 x y) → M_1 x.

(*  A13 : Il est possible qu'un Dieu existe *)
Axiom A13 : M(∃ x:U , G_1 x).

(*  A14 : x existe nécessairement si et seulement si x n'est pas fini *)
Axiom A14 : ∀ x:U,
  N(∃ y:U , y = x) ↔ ¬ F_1 x.

(*  A15 : Axiome sur le rapport entre existence temporelle et non-existence *)
(*  Note: Cette formulation évite d'utiliser la notation temporelle "at t" non définie *)
Axiom A15 : ∀ x:U,
  ¬G_1 x → (¬L(∃ y:U , y = x) ∧ N(∃ y:U , y = x) ↔ E_1 x).

(*  A16 : Si x et y ont un attribut commun, alors ils ont quelque chose en commun *)
Axiom A16 : ∀ x y:U,
  (∃ z:U , A_2 z x ∧ A_2 z y) → (∃ z:U , C_3 z x y).

(*  A17a-d : Axiomes sur les types d'attributs *)
Axiom A17a : ∀ x:U, U_1 x → ¬A_1 x.
Axiom A17b : ∀ x:U, W_1 x → ¬A_1 x.
Axiom A17c : ∀ x:U, D_1 x → ¬A_1 x.
Axiom A17d : ∀ x:U, J_1 x → ¬A_1 x.

(*  A18 : Si x et y sont des substances, alors x a plus de réalité que y ssi x a plus d'attributs que y

Axiom A18 : ∀ x y:U,
  (S_1 x ∧ S_1 y) → (R_2 x y ↔ V_2 x y).

(*  A19 : Si x est un attribut de y et est conçu par y et est en y, alors x est la puissance de y *)

Axiom A19 : ∀ x y:U,
  (((I_2 x y ∧ C_2 x y) ∧ I_2 y x) ∧ C_2 y x) = P_2 x y.

(*  ====PROPOSITIONS==== *)

(*  P1 : Si x est un mode de y et y est une substance, alors x est en y et y est en soi *)

Theorem P1 : ∀ x y:U,
  M_2 x y ∧ S_1 y → I_2 x y ∧ I_2 y y.
Proof.
  (*  On pose nos hypothèses *)
  intros x y [HM HS].

  (*  Premièrement, de HS (y est une substance) et D3 (définition de substance),
```

```
    on peut déduire que y est en soi *)
  assert (HIyy: I_2 y y).
  { apply D3 in HS. (*  Par D3 : y substance <-> y en soi et y conçu par soi *)
    destruct HS as [HIyy _]. (*  On garde juste y en soi *)
    exact HIyy.
  }

  (*  Deuxièmement, de HM (x est un mode de y) et D5a (définition de mode),
      on peut déduire que x est en y *)
  assert (HIxy: I_2 x y).
  { apply D5a in HM. (*  Par D5a : x mode de y <-> x\u2260y et x en y et x conçu par y *)
    destruct HM as [_ [HIxy _]]. (*  On garde juste x en y *)
    exact HIxy.
  }

  (*  Enfin, on combine les deux résultats *)
  split.
  - exact HIxy. (*  x est en y *)
  - exact HIyy. (*  y est en soi *)
Qed.

Theorem P2 : ∀ x y:U,
  S_1 x ∧ S_1 y ∧ x ≠ y → ¬ ∃ z:U, C_3 z x y.
Proof.
  (*  Introduction des hypothèses *)
  intros x y [HSx [HSy Hxy]].

  (*  Application de D3 à x *)
  assert (HCx: I_2 x x ∧ C_2 x x).
  { apply D3. exact HSx. }
  destruct HCx as [HIxx HCxx].

  (*  Application de D3 à y *)
  assert (HCy: I_2 y y ∧ C_2 y y).
  { apply D3. exact HSy. }
  destruct HCy as [HIyy HCyy].

  (*  Pour montrer ~C_2 x y *)
  assert (HNoCxy: ¬C_2 x y).
  { apply A2 in HCxx.
    intros HC2xy.
    apply HCxx.
    ∃ y; auto. }

  (*  Pour montrer ~C_2 y x *)
  assert (HNoCyx: ¬C_2 y x).
  { apply A2 in HCyy.
    intros HC2yx.
    apply HCyy.
    ∃ x; auto. }

  (*  Conclusion par A5 *)
  apply A5.
  split.
```

```
  - exact HNoCxy.
  - exact HNoCyx.
Qed.
(*  P3 : Si x et y n'ont rien en commun, alors x ne peut pas être cause de y et y ne peut pas être cause

Theorem P3 : ∀ x y:U,
  (¬ ∃ z:U, C_3 z x y) → ¬ K_2 x y ∧ ¬ K_2 y x.
Proof.
  (*  Introduction des hypothèses *)
  intros x y H_no_common.
  (*  Application de A5 à l'hypothèse :
     si x,y n'ont rien en commun alors ils ne peuvent être conçus l'un à travers l'autre *)

  apply A5 in H_no_common.
  destruct H_no_common as [H_no_Cxy H_no_Cyx].
  (*  Nous allons prouver les deux parties de la conjonction *)
  split.
  (*  Première partie : ~K_2 x y *)
  - intros H_Kxy.
    (*  Par A4, si x est cause de y alors y est conçu à travers x *)
    apply A4 in H_Kxy.
    (*  Contradiction avec H_no_Cyx *)
    contradiction.
  (*  Deuxième partie : ~K_2 y x *)
  - intros H_Kyx.
    (*  Par A4, si y est cause de x alors x est conçu à travers y *)
    apply A4 in H_Kyx.
    (*  Contradiction avec H_no_Cxy *)
    contradiction.
Qed.
   x neq y -> y neq x  Lemma neq_sym : ∀ x y:U,
  x ≠ y → y ≠ x.
Proof.
  intros x y H.
  intro Heq.
  apply H.
  symmetry.
  exact Heq.
Qed.
(*  x est une substance ssi x est en soi *)
Lemma DP1 : ∀ x:U,
  S_1 x ↔ I_2 x x.
Proof.
  intros. split.
  - intro. apply D3 in H. destruct H. exact H.
  - intro. apply D3. split.
    + assumption.
    + apply A8. assumption.
Qed.
```

DP4: Une substance est sa propre cause Lemma DP4 : ∀ x:U,
```
  S_1 x -> K_2 x x.
Proof.
  intros x HS.
  apply A4.
  apply D3 in HS.
  destruct HS as [_ HC].
  exact HC.
Qed.
```
DP5 : Toute chose est soit une substance soit un mode Lemma DP5 : ∀ x:U,
```
  S_1 x \/ M_1 x.
Proof.
  intro. pose proof (A1 x). destruct H.
  - left. apply DP1. assumption.
  - right. destruct H as [y [H1 H2]].
    pose proof (A8 _ _ H2). apply A12.
    exists y. apply D5a. split.
    + apply neq_sym. assumption.
    + split; assumption.
Qed.
```
DP6 : Une substance et un mode ne peuvent jamais être la même chose Lemma DP6 : ∀ x:U,
```
  ~(S_1 x /\ M_1 x).
Proof.
  intros x [HS HM].
  (*  x est un mode donc il existe une substance y dont x est le mode (D5b) *)
  apply D5b in HM. destruct HM as [y [HSy HMxy]].
  (*  x est un mode de y donc conçu par y (D5a) *)
  apply D5a in HMxy. destruct HMxy as [Hneq [_ HCxy]].
  (*  x est une substance donc conçu par soi (D3) *)
  apply D3 in HS. destruct HS as [_ HCxx].
  (*  Par A2, si x est conçu par soi, il ne peut être conçu par autre chose *)
  apply A2 in HCxx.
  apply HCxx.
  exists y.
  split.
  - apply neq_sym in Hneq. exact Hneq.
  - exact HCxy.
Qed.

(*  DP7 : Si x est un attribut de y et y est une substance, alors x = y *)
Lemma DP7 : forall x y:U, A_2 x y /\ S_1 y -> x = y.
Proof.
  (*  Introduction des hypothèses *)
  intros x y [HA2 HS].

  (*  De HA2 (x est un attribut de y) et D4b, on obtient que y est conçu à travers x *)
  apply D4b in HA2.
  destruct HA2 as [_ HCyx].

  (*  De HS (y est une substance) et D3, on obtient que y est conçu par soi *)
  apply D3 in HS.
  destruct HS as [_ HCyy].

  (*  Par A2, si y est conçu par soi, il ne peut être conçu par autre chose *)
```

```
  apply A2 in HCyy.
  (* Preuve par l'absurde : supposons x \u2260 y *)
  assert (H: x = y).
  { (* Par le tiers exclu, soit x = y soit x \u2260 y *)
    apply NNPP. (* Not Not P -> P *)
    intro Hneq.
    (* Si x \u2260 y, alors y serait conçu à travers quelque chose d'autre que soi *)
    apply HCyy.
    ∃ x.
    split.
    - exact Hneq.
    - exact HCyx.
  }
  exact H.
Qed.
```

DPI: Tout est soit une substance, soit un mode, mais pas les deux `Theorem DPI : ∀ x:U, (S_1 x ∧ ¬M_1 x) ∨ (¬S_1 x ∧ M_1 x).`

```
Proof.
  intro x. pose proof DP5 x.
  pose proof DP6 x. destruct H; [left | right];
  split; auto; intro; apply H0; auto; split; auto.
Qed.
```

DPII: Une substance est ses propres attributs `Theorem DPII : ∀ x:U, S_1 x → A_2 x x.`

```
Proof.
  intros x HS.
  (* Pour montrer A_2 x x, on utilise D4b *)
  apply D4b. split.
  - (* Montrons que x est un attribut (A_1 x) *)
    apply D4a.
    ∃ x.
    (* x est une substance (hypothèse HS) *)
    split; [exact HS||].
    (* De HS et D3, on obtient que x est en soi et conçu par soi *)
    apply D3 in HS as [HIxx HCxx].
    repeat split; assumption.
  - (* Montrons que x est conçu à travers x (C_2 x x) *)
    apply D3 in HS.
    destruct HS as [_ HCxx].
    exact HCxx.
Qed.
```

DPIII: Quelque chose est une substance ssi elle est causa sui `Theorem DPIII : ∀ x:U, S_1 x ↔ K_2 x x.`

```
Proof.
  intros. split; intro.
  - (* Si x est une substance, alors x est cause de soi *)
    (* Déjà prouvé par DP4 *)
    apply DP4. auto.
  - (* Si x est cause de soi, alors x est une substance *)
    (* Par A4, si x est cause de y, alors y est conçu à travers x *)
```

```
apply A4 in H. (*  Cela donne C_2 x x *)
(*  Par A1, tout ce qui est, est soit en soi, soit en autre chose *)
pose proof (A1 x) as [HIxx | HIxy].
+ (*  Si x est en soi (I_2 x x), alors par D3, x est une substance *)
  apply D3. split.
  × exact HIxx.
  × exact H.
+ (*  Si x est en autre chose (I_2 x y), alors... *)
  destruct HIxy as [y [Hneq HIxy]].
  (*  Par A8, si x est en y alors x est conçu par y *)
  apply A8 in HIxy.
  (*  Par A2, si x est conçu par soi, alors il n'est pas conçu par autre chose *)
  assert (HC: ¬ ∃ z : U , z ≠ x ∧ C_2 x z).
  { apply A2. exact H. }
  (*  Contradiction: x est conçu par y (y\u2260x) *)
  destruct HC. ∃ y. split; assumption.
Qed.
```

P4: Deux ou plusieurs choses distinctes ne peuvent se distinguer que par la diversité des attributs de leurs substances, ou par la diversité des affections de ces mêmes substances. `Theorem P4 : ∀ x y:U,`

```
  x ≠ y → ∃ z z':U ,
  ((A_2 z x ∧ A_2 z' y ∧ z ≠ z') ∨
   (A_2 z x ∧ z = x ∧ M_1 y) ∨
   (A_2 z' y ∧ z' = y ∧ M_1 x) ∨
   (M_1 x ∧ M_1 y)).
Proof.
  intros x y H.
  pose proof (DPI x) as [H0 | H0].
  - (*  x is substance *)
    pose proof (DPI y) as [H1 | H1].
    + (*  Both are substances *)
      ∃ x; ∃ y.
      left. split.
      × apply DPII. destruct H0. auto.
      × split.
        – apply DPII. destruct H1. auto.
        – auto.
    + (*  x is substance, y is mode *)
      ∃ x; ∃ y.
      right. left. split.
      × apply DPII. destruct H0. auto.
      × split.
        – reflexivity.
        – destruct H1. auto.
  - (*  x is mode *)
    pose proof (DPI y) as [H1 | H1].
    + (*  y is substance, x is mode *)
      ∃ x; ∃ y.
      right. right. left. split.
      × apply DPII. destruct H1. auto.
      × split.
        – reflexivity.
```

```
      – destruct H0. auto.
    + (*  Both are modes *)
      ∃ x; ∃ y.
      right. right. right.
      destruct H0. destruct H1. split; auto.
Qed.

(*  P5: Il ne peut y avoir, dans la nature des choses,
   deux ou plusieurs substances de même nature, ou,
   en d'autres termes, de même attribut. *)
(*  Prémisses: P4, DP6, DP7 ou DP7 seul ou P2, A16 seul *)
Theorem P5 : ∀ x y:U,
  S_1 x ∧ S_1 y ∧ x ≠ y → ¬ ∃ z:U, (A_2 z x ∧ A_2 z y).
Proof.
  (*  Introduction des hypothèses *)
  intros x y [HSx [HSy Hneq]].

  (*  Preuve par contradiction *)
  intro H.
  destruct H as [z H].
  destruct H as [HAzx HAzy].

  (*  Si z est un attribut de x et de y, alors par DP7, z = x et z = y *)
  assert (Hz_eq_x: z = x).
  { apply DP7. split; assumption. }

  assert (Hz_eq_y: z = y).
  { apply DP7. split; assumption. }

  (*  Par transitivité de l'égalité, x = y *)
  assert (Hx_eq_y: x = y).
  { rewrite ← Hz_eq_x. exact Hz_eq_y. }

  (*  Contradiction avec l'hypothèse x \u2260 y *)
  contradiction.
Qed.

(*  P6: Une substance ne peut être produite par une autre substance. *)
(*  Prémisses: P2, P3 *)
Theorem P6 : ∀ x y:U,
  S_1 x ∧ S_1 y ∧ x ≠ y → ~(K_2 x y ∧ ¬K_2 y x).
Proof.
  (*  Introduction des hypothèses *)
  intros x y [HSx [HSy Hneq]].

  (*  Preuve par contradiction *)
  intro H.
  destruct H as [HKxy HNKyx].

  (*  Par P2, deux substances n'ont rien en commun *)
  assert (HNoCommon: ¬ ∃ z:U, C_3 z x y).
  { apply P2. split; [exact HSx | split; [exact HSy | exact Hneq]]. }

  (*  Par P3, si deux choses n'ont rien en commun, l'une ne peut pas être cause de l'autre *)

  assert (HNoCauses: ¬ K_2 x y ∧ ¬ K_2 y x).
```

```
  { apply P3. assumption. }

  (* La contradiction est évidente : HKxy contredit la première partie de HNoCauses *)
  destruct HNoCauses as [HNKxy _].
  contradiction.
Qed.

(* P6c: Corollaire de P6 *)
(* Prémisses: D3, A2, A4 *)
Corollary P6c : ∀ x:U,
  S_1 x → ~(∃ y:U, y ≠ x ∧ K_2 y x).
Proof.
  (* Introduction des hypothèses *)
  intros x HSx.

  (* Preuve par contradiction *)
  intro H.
  destruct H as [y [Hneq HKyx]].

  (* Par A4, si y est cause de x, alors x est conçu à travers y *)
  assert (HCxy: C_2 x y).
  { apply A4. exact HKyx. }

  (* Par D3, si x est une substance, alors x est conçu par soi *)
  assert (HCxx: C_2 x x).
  { apply D3 in HSx. destruct HSx as [_ HCxx]. exact HCxx. }

  (* Par A2, si x est conçu par soi, il ne peut être conçu par une autre chose *)
  assert (H_no_Cxy: ¬ ∃ z : U, z ≠ x ∧ C_2 x z).
  { apply A2. exact HCxx. }

  (* Contradiction: x est conçu à travers y (y\u2260x) *)
  apply H_no_Cxy. ∃ y. split; assumption.
Qed.

(* P7: Il appartient à la nature de la substance d'exister. *)
(* Prémisses: D3, P6c, A4, D1 *)
Theorem P7 : ∀ x:U,
  S_1 x → L(∃ y:U, y = x).
Proof.
  (* Introduction des hypothèses *)
  intros x HSx.

  (* Par D3, une substance est ce qui est conçu par soi *)
  assert (HCxx: C_2 x x).
  { apply D3 in HSx. destruct HSx as [_ HCxx]. exact HCxx. }

  (* Par A4, si x est conçu par x, alors x est cause de x *)
  assert (HKxx: K_2 x x).
  { apply A4. exact HCxx. }

  (* Par P6c, si x est une substance, alors aucune autre chose ne peut être cause de x *)

  assert (HNoExternalCause: ~(∃ y:U, y ≠ x ∧ K_2 y x)).
  { apply P6c. exact HSx. }
```

```
  (*  Donc x est sa propre cause et n'a pas de cause externe *)
  assert (H_causa_sui: K_2 x x ∧ ¬ ∃ y:U, y ≠ x ∧ K_2 y x).
  { split; assumption. }

  (*  Par D1, une chose est cause de soi-même ssi son essence implique son existence *)
  apply D1. exact H_causa_sui.
Qed.

(*  P8: Toute substance est nécessairement infinie. *)
(*  Prémisses: D2, A9, A11, P5 *)
Theorem P8 : ∀ x:U,
  S_1 x → ¬F_1 x.
Proof.
  (*  Introduction des hypothèses *)
  intros x HSx.
  (*  Preuve par contradiction *)
  intro HFx.
  (*  Par D2, une chose est finie quand elle peut être limitée par une autre de même nature *)

  apply D2 in HFx.
  destruct HFx as [y [Hneq [HLyx HSameAttr]]].
  (*  Par A11, si x est une substance et y limite x, alors y est une substance *)
  assert (HSy: S_1 y).
  { apply A11 with (x := x). split; assumption. }

  (*  Par A9, toute chose a au moins un attribut *)
  assert (H_has_attr_x: ∃ z:U, A_2 z x).
  { apply A9. }
  destruct H_has_attr_x as [z HA2zx].
  (*  Par HSameAttr, z est aussi un attribut de y *)
  assert (HA2zy: A_2 z y).
  { apply HSameAttr. exact HA2zx. }

  (*  Par P5, deux substances distinctes ne peuvent pas avoir le même attribut *)
  assert (H_no_common_attr: ¬ ∃ z:U, (A_2 z x ∧ A_2 z y)).
  { apply P5. split; [exact HSx | split; [exact HSy | apply neq_sym; exact Hneq]]. }

  (*  Contradiction: z est un attribut commun à x et y *)
  apply H_no_common_attr. ∃ z. split; assumption.
Qed.

(*  P9: Suivant qu'une chose a plus de réalité ou d'être,
   un plus grand nombre d'attributs lui appartient. *)
(*  Prémisses: A18 - Relation entre réalité et attributs *)
Theorem P9 : ∀ x y:U,
  (S_1 x ∧ S_1 y) → (R_2 x y ↔ V_2 x y).
Proof.
  (*  Utilisation directe de l'axiome A18 *)
  intros x y H.
  apply A18.
  exact H.
```

```
Qed.
(*  P10: Tout attribut d'une substance doit être conçu par soi. *)
(*  Prémisses: D3, D4a, A2 *)
Theorem P10 : ∀ x:U,
  A_1 x → C_2 x x.
Proof.
  (*  Introduction des hypothèses *)
  intros x HA1.
  (*  Par D4a, si x est un attribut, alors il existe une substance y telle que... *)
  apply D4a in HA1.
  destruct HA1 as [y [HSy [HIxy [HCxy [HIyx HCyx]]]]].
  (*  Par D3 et HSy, y est conçu par soi *)
  assert (HCyy: C_2 y y).
  { apply D3 in HSy. destruct HSy as [_ HCyy]. exact HCyy. }

  (*  Par A2, si y est conçu par soi, il ne peut être conçu par autre chose que soi *)
  assert (H_no_Cyz: ¬ ∃ z : U , z ≠ y ∧ C_2 y z).
  { apply A2. exact HCyy. }

  (*  De HCyx, y est conçu à travers x *)
  (*  Si x \u2260 y, cela contredirait H_no_Cyz *)
  assert (Hxy_eq: x = y).
  { (*  Par le tiers exclu, soit x = y soit x \u2260 y *)
    apply NNPP. (*  Not Not P -> P *)
    intro Hneq.
    (*  Si x \u2260 y, alors y serait conçu à travers quelque chose d'autre que soi *)
    apply H_no_Cyz. ∃ x.
    split; assumption.
  }

  (*  Si x = y, alors nous pouvons substituer x pour y dans C_2 y y *)
  subst y.
  exact HCyy.
Qed.
(*  P11: Dieu, c'est-à-dire une substance constituée par une infinité d'attributs
   dont chacun exprime une essence éternelle et infinie, existe nécessairement. *)
(*  Prémisses: D1, D3, D4a, D4b, D6, A1, A2, A4, A8, A13, A7, P7 *)
Theorem P11 : L(∃ x:U , G_1 x).
Proof.
  (*  Approche: utiliser l'argument ontologique modal de la règle S5:
     Si (1) P est possible et (2) P implique nécessairement que P est nécessaire,
     alors P est nécessaire *)
  (*  Étape 1: Montrer que "un Dieu existe" est possible, directement par A13 *)
  pose proof A13 as H_God_possible.
  (*  Étape 2: Montrer que "un Dieu existe" implique nécessairement que "un Dieu existe" est nécessaire

  (*  Nous allons démontrer que : L((exists x:U, G_1 x) -> L(exists x:U, G_1 x)) *)
  (*  D'abord, prouvons que : (exists x:U, G_1 x) -> L(exists x:U, G_1 x) *)
  assert (H_God_to_nec: (∃ x:U , G_1 x) → L(∃ x:U , G_1 x)).
  {
```

```
    intros H_God_exists.
    destruct H_God_exists as [g H_g_is_God].
    (*  Par D6, Dieu est une substance *)
    assert (H_g_is_substance: S_1 g).
    { apply D6 in H_g_is_God. destruct H_g_is_God. exact H. }

    (*  Par P7, toute substance existe nécessairement *)
    assert (H_g_exists_necessarily: L(∃ y:U , y = g)).
    { apply P7. exact H_g_is_substance. }

    (*  Nous devons maintenant transformer L(exists y:U, y = g) en L(exists x:U, G_1 x) *)

    (*  Nous savons que g est Dieu, donc ce qui existe nécessairement est un Dieu *)

    (*  Étape intermédiaire: "g existe" implique "un Dieu existe" *)
    assert (H_g_to_God: (∃ y:U , y = g) → (∃ x:U , G_1 x)).
    {
      intros H_g_exists.
      ∃ g.
      exact H_g_is_God.
    }

    (*  Par R5 (règle de nécessitation), nous pouvons transformer cela en nécessité *)
    assert (H_nec_impl: L((∃ y:U , y = g) → (∃ x:U , G_1 x))).
    { apply R5. exact H_g_to_God. }

    (*  Par R3 (axiome K), nous distribuons la nécessité sur l'implication *)
    assert (H_impl_nec: L(∃ y:U , y = g) → L(∃ x:U , G_1 x)).
    { apply R3. exact H_nec_impl. }

    (*  Finalement, nous appliquons l'implication à notre hypothèse *)
    apply H_impl_nec.
    exact H_g_exists_necessarily.
  }

  (*  Maintenant, appliquons R5 (règle de nécessitation) pour obtenir:
     L((exists x:U, G_1 x) -> L(exists x:U, G_1 x)) *)
  assert (H_nec_God_to_nec: L((∃ x:U , G_1 x) → L(∃ x:U , G_1 x))).
  { apply R5. exact H_God_to_nec. }

  (*  Étape 3: Appliquer la règle ontologique (Mp & L(p -> Lp)) -> Lp *)
  (*  Avec p = (exists x:U, G_1 x) *)
  apply ontological_rule.
  - (*  M(exists x:U, G_1 x) - Possibilité de l'existence de Dieu *)
    exact H_God_possible.
  - (*  L((exists x:U, G_1 x) -> L(exists x:U, G_1 x)) - Nécessité de l'implication *)
    exact H_nec_God_to_nec.
Qed.

(*  P12: On ne peut concevoir selon sa véritable nature aucun attribut de la substance
   duquel il résulte que la substance soit divisible. *)
(*  Prémisses: A10, P7 *)
```

```
Theorem P12 : ∀ x y z:U,
  S_1 x → ¬D_3 x y z.
Proof.
  (* Introduction des hypothèses *)
  intros x y z HS.

  (* Preuve par contradiction *)
  intro HD.

  (* Par A10, si x est divisible en y et z, alors il est possible que x n'existe pas *)
  assert (HMnot_exists: M(¬ ∃ w:U , w = x)).
  { apply A10 with (y:=y) (z:=z). exact HD. }

  (* Par P7, si x est une substance, alors x existe nécessairement (son essence implique son existence)

  assert (HL_exists: L(∃ w:U , w = x)).
  { apply P7. exact HS. }

  (* Par A7, possibilité de non-existence et nécessité logique sont contradictoires *)
  (* Si p est nécessaire (L(p)), alors ~p ne peut pas être possible (M(~p)) *)
  assert (HNot_M_not_exists: ¬M(¬ ∃ w:U , w = x)).
  {
    (* Utilisation d'A7: M(~p) <-> ~L(p) *)
    (* Si nous avons L(p), alors ~M(~p) par contraposition *)
    (* Prouvons la contraposée de A7 *)
    assert (H_contra_A7: L(∃ w:U , w = x) → ¬M(¬ ∃ w:U , w = x)).
    {
      intro HLexists.
      intro HMnotExists.
      (* Par A7, M(~p) <-> ~L(p) *)
      apply A7 in HMnotExists.
      (* Donc ~L(p), mais nous savons L(p), contradiction *)
      contradiction.
    }
    (* Application de la contraposée *)
    apply H_contra_A7.
    exact HL_exists.
  }

  (* Contradiction finale: nous avons à la fois M(~p) et ~M(~p) *)
  contradiction.
Qed.

(* P13: La substance absolument infinie est indivisible. *)
(* Prémisses: P12 *)
Theorem P13 : ∀ x y z:U,
  (S_1 x ∧ (∀ w:U, A_1 w → A_2 w x)) → ¬D_3 x y z.
Proof.
  (* Introduction des hypothèses *)
  intros x y z [HS HAllAttributes].

  (* Application directe de P12 *)
  (* P12 affirme que toute substance est indivisible *)
  (* Puisque x est une substance (hypothèse HS),
```

```
    nous pouvons directement appliquer P12 *)
  apply P12.
  exact HS.
Qed.

(* P14: Il ne peut exister et on ne peut concevoir aucune autre substance que Dieu. *)
(* Prémisses: P11, D6, A9, D4b, et soit DP7 soit P5 *)
Theorem P14 : ∃ x:U,
  G_1 x ∧ (∀ y:U, S_1 y → y = x).
Proof.
  (* Étape 1: Montrer qu'un Dieu existe *)
  (* Par P11, on sait que Dieu existe nécessairement: L(exists x:U, G_1 x) *)
  (* Par DR1, on peut déduire qu'un Dieu existe effectivement: exists x:U, G_1 x *)
  assert (HGodExists: ∃ x:U, G_1 x).
  {
    (* Appliquons DR1 (L(P) -> P) à P11 *)
    apply DR1.
    exact P11.
  }

  (* Étape 2: Extraire le Dieu g qui existe *)
  destruct HGodExists as [g HGg].
  (* Étape 3: Construire notre témoin, g, pour l'existentiel *)
  ∃ g.
  split.
  - (* Première partie: g est Dieu *)
    exact HGg.
  - (* Seconde partie: Toute substance est identique à g *)
    intros y HSy.
    (* Par D6, g est une substance possédant tous les attributs *)
    assert (HSg: S_1 g).
    { apply D6 in HGg. destruct HGg. exact H. }

    (* Par A9, toute substance possède au moins un attribut *)
    assert (HAttrY: ∃ a:U, A_2 a y).
    { apply A9. }
    destruct HAttrY as [a HAay].
    (* Par D6, g est une substance possédant tous les attributs,
       donc a est aussi un attribut de g *)
    assert (HAag: A_2 a g).
    {
      (* Par D6, g possède tous les attributs *)
      assert (HAllAttr: ∀ w:U, A_1 w → A_2 w g).
      { apply D6 in HGg. destruct HGg. exact H0. }

      (* De A_2 a y, on déduit A_1 a par D4b *)
      assert (HA1a: A_1 a).
      { apply D4b in HAay. destruct HAay. exact H. }

      (* Donc a est un attribut de g *)
      apply HAllAttr.
```

```
      exact HA1a.
    }

    (*  Nous avons maintenant deux substances, g et y, qui partagent le même attribut a *)

    (*  Par P5, deux substances ne peuvent pas partager le même attribut, donc g = y *)

    (*  Préparation pour appliquer P5 par contraposition *)
    (*  P5 affirme: S_1 x /\ S_1 y /\ x <> y -> ~ exists z:U, (A_2 z x /\ A_2 z y) *)
    (*  Par contraposition: exists z:U, (A_2 z x /\ A_2 z y) -> ~(S_1 x /\ S_1 y /\ x <> y) *)

    (*  Ou de manière équivalente: exists z:U, (A_2 z x /\ A_2 z y) /\ S_1 x /\ S_1 y -> x = y *)


    assert (HSameSubstance: g = y).
    {
      (*  Preuve par contradiction *)
      apply NNPP. (*  Not Not P -> P *)
      intro Hneq.
      (*  Si g \u2260 y, alors nous avons deux substances différentes avec le même attribut *)

      (*  Ce qui contredit P5 *)
      assert (HContra: ¬ ∃ z:U, (A_2 z g ∧ A_2 z y)).
      { apply P5. split; [exact HSg | split; [exact HSy | exact Hneq]]. }

      (*  Mais nous savons qu'il existe un attribut a tel que A_2 a g /\ A_2 a y *)
      apply HContra.
      ∃ a.
      split; assumption.
    }

    (*  Donc y = g *)
    symmetry. exact HSameSubstance.
Qed.
(*  P14-A: Version alternative de P14 *)
(*  Prémisses: P14, D6 *)
Theorem P14_A : ∃ x:U,
  ∀ y:U, G_1 y ↔ y = x.
Proof.
  (*  Par P14, il existe un Dieu g unique *)
  destruct P14 as [g [HGg HUnique]].
  (*  Ce Dieu g est notre témoin *)
  ∃ g.
  (*  Pour tout y, y est Dieu si et seulement si y = g *)
  intros y.
  split.
  (*  Première direction: si y est Dieu, alors y = g *)
  - intro HGy.
    (*  Par D6, y est une substance *)
    assert (HSy: S_1 y).
```

```
    { apply D6 in HGy. destruct HGy. exact H. }
    (*  Par HUnique, toute substance est identique à g *)
    apply HUnique.
    exact HSy.
  (*  Seconde direction: si y = g, alors y est Dieu *)
  - intro Heq.
    (*  Si y = g et g est Dieu, alors y est Dieu *)
    rewrite Heq.
    exact HGg.
Qed.
(*  P15: Tout ce qui existe est en Dieu et rien ne peut être ni être conçu sans Dieu *)
(*  Prémisses: DP5, P14, D3, D5b, D5a *)
Theorem P15 : ∀ x:U,
  ∃ g:U, G_1 g ∧ I_2 x g ∧ C_2 x g.
Proof.
  intro x.
  (*  Par P14, il existe un Dieu unique g *)
  destruct P14 as [g [HGg HGodUnique]].
  ∃ g.
  (*  Montrons que g est Dieu, que x est en g et que x est conçu par g *)
  split; [exact HGg|].
  (*  Par DP5, tout objet est soit une substance, soit un mode *)
  assert (H_substance_or_mode: S_1 x ∨ M_1 x).
  { apply DP5. }

  (*  Cas 1: x est une substance *)
  destruct H_substance_or_mode as [HSx | HMx].
  {
    (*  Si x est une substance, alors par l'unicité de Dieu (P14), x = g *)
    assert (Hxg: x = g).
    { apply HGodUnique. exact HSx. }

    (*  Par D3, une substance est en soi et conçue par soi *)
    assert (H_in_and_conceived: I_2 x x ∧ C_2 x x).
    { apply D3. exact HSx. }

    (*  Puisque x = g, I_2 x g <-> I_2 x x et C_2 x g <-> C_2 x x *)
    split.
    - (*  x est en g (car x = g et une substance est en soi) *)
      destruct H_in_and_conceived as [HIxx _].
      (*  Nous devons prouver I_2 x g, mais nous avons I_2 x x et x = g *)
      assert (I_2x_eq: I_2 x x → I_2 x g).
      { intro H. rewrite ← Hxg. exact H. }
      apply I_2x_eq. exact HIxx.
    - (*  x est conçu par g (car x = g et une substance est conçue par soi) *)
      destruct H_in_and_conceived as [_ HCxx].
      (*  Nous devons prouver C_2 x g, mais nous avons C_2 x x et x = g *)
      assert (C_2x_eq: C_2 x x → C_2 x g).
      { intro H. rewrite ← Hxg. exact H. }
      apply C_2x_eq. exact HCxx.
```

```
  }

  (* Cas 2: x est un mode *)
  {
    (* Par D5b, si x est un mode, alors il existe une substance y dont x est le mode *)
    apply D5b in HMx.
    destruct HMx as [y [HSy HMxy]].
    (* Par l'unicité de Dieu (P14), y = g *)
    assert (Hyg: y = g).
    { apply HGodUnique. exact HSy. }

    (* Substituer g pour y dans M_2 x y *)
    rewrite Hyg in HMxy.
    (* Par D5a, si x est un mode de g, alors x est en g et conçu par g *)
    apply D5a in HMxy.
    destruct HMxy as [_ [HIxg HCxg]].
    split; assumption.
  }
Qed.
(* P16: Dieu est cause de toutes choses *)
(* Prémisses: P15, A4 *)
Theorem P16 : ∀ x:U,
  ∃ g:U , G_1 g ∧ K_2 g x.
Proof.
  intro x.
  (* Par P15, tout ce qui existe est en Dieu et conçu par Dieu *)
  pose proof (P15 x) as [g [HGg [HIxg HCxg]]].
  (* Notre témoin est g (Dieu) *)
  ∃ g.
  (* Première partie: g est Dieu *)
  split; [exact HGg|].
  (* Seconde partie: g est cause de x *)
  (* Par A4, si x est conçu par g, alors g est cause de x *)
  apply A4.
  exact HCxg.
Qed.
(* P17: Dieu agit par les seules lois de sa nature *)
(* Prémisses: P14, P15, P16, D7a *)
Theorem P17 : ∃ g:U ,
  G_1 g ∧ ~(∃ x:U , ¬I_2 x g ∧ K_2 x g) ∧ (∀ x:U, K_2 g x).
Proof.
  (* Par P14, il existe un Dieu unique g *)
  destruct P14 as [g [HGg HGodUnique]].
  ∃ g.
  (* Nous devons prouver 3 choses:
     1. g est Dieu
     2. g n'est déterminé à agir par aucune cause externe
     3. g est cause de toutes choses *)
  split; [exact HGg||].
```

```
(*  Montrons que g n'est déterminé à agir par aucune cause externe *)
split.
{
  (*  Preuve par contradiction *)
  intro HExternal.
  destruct HExternal as [x [HNotInG HCauseG]].
  (*  Par D6, g est une substance *)
  assert (HSg: S_1 g).
  { apply D6 in HGg. destruct HGg. exact H. }

  (*  Par P6c, aucune chose distincte ne peut être cause d'une substance *)
  assert (HNoCause: ~(∃ y:U , y ≠ g ∧ K_2 y g)).
  { apply P6c. exact HSg. }

  (*  Pour établir la contradiction, montrons que x \u2260 g *)
  assert (Hxg: x ≠ g).
  {
    (*  Preuve par contradiction *)
    intro Heq.
    (*  Si x = g, alors "x n'est pas en g" signifierait "g n'est pas en g" *)
    rewrite Heq in HNotInG.
    (*  Mais par D3, une substance est en soi *)
    assert (HIgg: I_2 g g).
    { apply D3 in HSg. destruct HSg. exact H. }
    (*  Contradiction avec HNotInG *)
    contradiction.
  }

  (*  Maintenant nous avons x \u2260 g et K_2 x g, ce qui contredit HNoCause *)
  apply HNoCause.
  ∃ x.
  split; assumption.
}

(*  Montrons que g est cause de toutes choses - conséquence directe de P16 *)
intro x.
assert (HGodCause: ∃ h:U , G_1 h ∧ K_2 h x).
{ apply P16. }
destruct HGodCause as [h [HGh HKhx]].
(*  Par P14_A, h = g car h est Dieu et g est le seul Dieu *)
assert (Hhg: h = g).
{
  destruct P14_A as [k HP14A].
  assert (Hhk: h = k).
  { apply HP14A. exact HGh. }
  assert (Hgk: g = k).
  { apply HP14A. exact HGg. }
  rewrite Hhk.
  symmetry.
  exact Hgk.
}
```

```
  (*  Donc g est cause de x *)
  rewrite ← Hhg.
  exact HKhx.
Qed.

(*  P17c2: Dieu seul est cause libre *)
(*  Prémisses: P17, D7a, P14 *)
Theorem P17c2 : ∃ g:U ,
  G_1 g ∧ B_1 g ∧ (∀ x:U, B_1 x → x = g).
Proof.
  (*  Par P17, il existe un Dieu g qui est cause de toutes choses et n'est déterminé par aucune cause ex

  destruct P17 as [g [HGg [HNoExternalCause HAllCauses]]].
  ∃ g.
  (*  Montrons les trois parties du théorème:
     1. g est Dieu
     2. g est libre
     3. Seul g est libre (toute cause libre est identique à g) *)
  (*  Première partie: g est Dieu *)
  split; [exact HGg||].
  (*  Deuxième partie: g est libre *)
  split.
  {
    (*  Par D7a, une chose est libre ssi elle est cause que d'elle-même *)
    apply D7a.
    (*  Par DPIII, une substance est cause de soi-même *)
    assert (HSg: S_1 g).
    { apply D6 in HGg. destruct HGg. exact H. }

    assert (HKgg: K_2 g g).
    { apply DPIII. exact HSg. }

    (*  Par HNoExternalCause, g n'est déterminé à agir par aucune cause externe *)
    (*  Donc g est sa propre cause et n'a pas de cause externe *)
    split.
    - (*  g est cause de soi *)
      exact HKgg.
    - (*  g n'a pas de cause externe *)
      (*  Preuve par contradiction *)
      intro HExternalCause.
      destruct HExternalCause as [y [Hyg HKyg]].
      (*  Si y est cause de g, alors par P15, y est en g *)
      pose proof (P15 y) as [h [HGh [HIyh HCyh]]].
      (*  Par P14_A, h = g car h est Dieu et g est le seul Dieu *)
      assert (Hhg: h = g).
      {
        destruct P14_A as [k HP14A].
        assert (Hhk: h = k).
        { apply HP14A. exact HGh. }
        assert (Hgk: g = k).
```

```
      { apply HP14A. exact HGg. }
      rewrite Hhk.
      symmetry.
      exact Hgk.
    }
    rewrite Hhg in HIyh.
    (*  Nous avons donc y en g (HIyg) et y cause de g (HKyg) *)
    (*  Mais par HNoExternalCause, il n'existe pas de x tel que ~I_2 x g /\ K_2 x g *)
    (*  Donc ~(~I_2 y g /\ K_2 y g), ce qui est logiquement équivalent à I_2 y g \/ ~K_2 y g *)

    (*  Puisque nous avons I_2 y g, il n'y a pas de contradiction *)
    (*  Pour obtenir une contradiction, montrons que y \u2260 g *)
    assert (Hyg': y ≠ g).
    {
      intro Heq.
      (*  Si y = g, nous n'aurions pas une cause externe *)
      rewrite Heq in Hyg.
      contradiction.
    }

    (*  Par P6c, aucune chose distincte ne peut être cause d'une substance *)
    assert (HNoCause: ~(∃ z:U, z ≠ g ∧ K_2 z g)).
    { apply P6c. exact HSg. }

    (*  Maintenant nous avons y \u2260 g et K_2 y g, ce qui contredit HNoCause *)
    apply HNoCause.
    ∃ y.
    split; assumption.
}

(*  Troisième partie: toute cause libre est identique à g *)
intros x HBx.
(*  Par D7a, si x est libre, alors x est cause de soi-même et n'a pas de cause externe *)

apply D7a in HBx.
destruct HBx as [HKxx HNoExternalX].
(*  Par HKxx et DPIII, x est une substance *)
assert (HSx: S_1 x).
{ apply DPIII. exact HKxx. }

(*  Par P14, toute substance est identique à g *)
destruct P14 as [h [HGh HGodUnique]].
(*  Nous devons montrer que h = g, puis que x = h, et donc x = g *)
assert (Hhg: h = g).
{
  destruct P14_A as [k HP14A].
  assert (Hhk: h = k).
  { apply HP14A. exact HGh. }
  assert (Hgk: g = k).
  { apply HP14A. exact HGg. }
```

```
    rewrite Hhk.
    symmetry.
    exact Hgk.
  }

  rewrite Hhg in HGodUnique.
  apply HGodUnique.
  exact HSx.
Qed.

(* P18: Dieu est cause immanente de toutes choses *)
(* Prémisses: P15, P16 *)
Theorem P18 : ∃ g:U,
  G_1 g ∧ (∀ x:U, I_2 x g ↔ K_2 g x).
Proof.
  (* Par P16, il existe un Dieu g qui est cause de toutes choses *)
  (* Choisissons n'importe quel x, pour démontrer l'existence de Dieu *)
  destruct P14 as [g [HGg _]].
  ∃ g.

  (* Montrons deux choses:
     1. g est Dieu
     2. Pour tout x, x est en g si et seulement si g est cause de x *)
  split; [exact HGg|].

  (* Pour tout objet x *)
  intro x.
  split.

  (* Première direction: si x est en g, alors g est cause de x *)
  - intro HIxg.
    (* Par A8, si x est en g, alors x est conçu par g *)
    assert (HCxg: C_2 x g).
    { apply A8. exact HIxg. }
    (* Par A4, si x est conçu par g, alors g est cause de x *)
    apply A4. exact HCxg.

  (* Seconde direction: si g est cause de x, alors x est en g *)
  - intro HKgx.
    (* Par P15, tout ce qui existe est en Dieu et conçu par Dieu *)
    destruct (P15 x) as [h [HGh [HIxh _]]].

    (* Puisque g et h sont tous deux Dieu, et par P14 il n'existe qu'un seul Dieu,
       g et h doivent être identiques *)
    assert (Hgh: g = h).
    { (* Par P14_A, il existe un unique Dieu *)
      destruct P14_A as [k HP14A].
      (* g est Dieu donc g = k *)
      assert (Hgk: g = k).
      { apply HP14A. exact HGg. }
      (* h est Dieu donc h = k *)
      assert (Hhk: h = k).
      { apply HP14A. exact HGh. }
      (* Par transitivité, g = h *)
      rewrite Hgk. symmetry. exact Hhk.
    }
```

```
    (* Substituons g pour h dans HIxh *)
    rewrite ← Hgh in HIxh.
    exact HIxh.
Qed.

(* P19: Dieu et tous ses attributs sont éternels *)
(* Prémisses: D8, P14, P14-A, D4b, P10, DP4 *)
Theorem P19 : ∃ g:U,
  G_1 g ∧ E_1 g ∧ (∀ x:U, A_2 x g → E_1 x).
Proof.
  (* Par P14, il existe un Dieu g unique *)
  destruct P14 as [g [HGg _]].
  ∃ g.
  (* Montrons trois choses:
     1. g est Dieu
     2. g est éternel
     3. Tous les attributs de g sont éternels *)
  (* Première partie: g est Dieu *)
  split; [exact HGg|].
  (* Deuxième partie: g est éternel *)
  split.
  {
    (* Par D6, g est une substance *)
    assert (HSg: S_1 g).
    { apply D6 in HGg. destruct HGg. exact H. }

    (* Par P7, si x est une substance, alors son essence implique son existence,
       i.e., il existe nécessairement *)
    assert (HNecExist: L(∃ y:U, y = g)).
    { apply P7. exact HSg. }

    (* Par D8, l'éternité est l'existence même en tant que nécessaire *)
    apply D8. exact HNecExist.
  }

  (* Troisième partie: Tous les attributs de g sont éternels *)
  intros x HA2xg.
  (* Par D4b, si x est un attribut de g, alors x est un attribut et g est conçu à travers x *)

  apply D4b in HA2xg.
  destruct HA2xg as [HA1x HCgx].
  (* Par P10, si x est un attribut, alors x est conçu par soi *)
  assert (HCxx: C_2 x x).
  { apply P10. exact HA1x. }

  (* Par DP7, si x est un attribut de g et g est une substance, alors x = g *)
  (* Commençons par montrer que g est une substance *)
  assert (HSg: S_1 g).
  { apply D6 in HGg. destruct HGg. exact H. }
```

```
  (*  Maintenant appliquons DP7 *)
  assert (Hxg: x = g).
  { apply DP7. split; [apply D4b; split; assumption | exact HSg]. }

  (*  Remplaçons x par g *)
  rewrite Hxg.
  (*  Nous avons déjà montré que g est éternel *)
  assert (HEg: E_1 g).
  { apply D8. apply P7. exact HSg. }

  exact HEg.
Qed.
(*  P20: L'existence et l'essence de Dieu sont une seule et même chose *)
(*  Prémisses: D4b, P10, DP4, P14, P14-A *)
Theorem P20 : ∀ x:U,
  ∃ g:U, G_1 g ∧ (A_2 x g → x = g).
Proof.
  intro x.
  (*  Par P14, il existe un Dieu unique g *)
  destruct P14 as [g [HGg HUnique]].
  ∃ g.
  (*  Montrons deux choses:
     1. g est Dieu
     2. Si x est un attribut de g, alors x = g *)
  (*  Première partie: g est Dieu *)
  split; [exact HGg|].
  (*  Deuxième partie: Si x est un attribut de g, alors x = g *)
  intro HA2xg.
  (*  Par D6, g est une substance *)
  assert (HSg: S_1 g).
  { apply D6 in HGg. destruct HGg. exact H. }

  (*  Par DP7, si x est un attribut de g et g est une substance, alors x = g *)
  apply DP7. split; assumption.
Qed.
(*  P21: Tout ce qui suit de l'essence absolue d'un attribut de Dieu existe nécessairement et infiniment

(*  Prémisses: P19, D8, A3, A14, R1, R6, R7, P20, DP7 *)
Theorem P21 : ∀ x:U,
  (∃ g y:U, G_1 g ∧ A_2 y g ∧ x ≠ g ∧ K_2 y x ∧ ~(∃ z:U, z ≠ y ∧ K_2 z x)) →
  (N(∃ v:U, v = x) ∧ ¬F_1 x).
Proof.
  intros x H.
  destruct H as [g [y [HGg [HA2yg [Hxg [HKyx HNoOtherCause]]]]]].
  (*  Par D6, si g est Dieu, alors g est une substance *)
  assert (HSg: S_1 g).
  { apply D6 in HGg. destruct HGg. exact H. }

  (*  Par DP7, si y est un attribut de g et g est une substance, alors y = g *)
```

```
assert (Hyg: y = g).
{ apply DP7. split; assumption. }

(*  Remplaçons y par g dans HKyx *)
rewrite Hyg in HKyx.
(*  Nous avons maintenant: HKyx : K_2 g x *)
(*  Par P19, g est éternel *)
assert (HEg: E_1 g).
{ destruct P19 as [h [HGh [HEh _]]].
  (*  Puisque g et h sont tous deux Dieu, g = h *)
  assert (Hgh: g = h).
  { destruct P14_A as [k HP14A].
    assert (Hgk: g = k).
    { apply HP14A. exact HGg. }
    assert (Hhk: h = k).
    { apply HP14A. exact HGh. }
    rewrite Hgk. symmetry. exact Hhk.
  }
  rewrite Hgh. exact HEh.
}

(*  Par D8, g existe nécessairement (logiquement) *)
assert (HLg: L(∃ v:U , v = g)).
{ apply D8. exact HEg. }

(*  Par R1, nécessité logique implique nécessité naturelle *)
assert (HNg: N(∃ v:U , v = g)).
{ apply R1. exact HLg. }

(*  Par A3, d'une cause déterminée suit nécessairement un effet *)
assert (HNec: N((∃ v:U , v = g) ↔ ∃ v:U , v = x)).
{ apply A3. exact HKyx. }

(*  Démontrons maintenant les deux parties du résultat *)
split.
- (*  1. x existe nécessairement *)
  (*  Utilisons HNec pour extraire la partie gauche de la double implication *)
  assert (HNecImpl1: N((∃ v:U , v = g) → (∃ v:U , v = x))).
  {
    (*  Utilisons R7 pour extraire l'implication à partir de l'équivalence *)
    apply R7. exact HNec.
  }

  (*  Appliquons R6 pour distribuer N sur l'implication *)
  (*  R6: N(P -> Q) -> (N(P) -> N(Q)) *)
  assert (HNecDist: N(∃ v:U , v = g) → N(∃ v:U , v = x)).
  { apply R6. exact HNecImpl1. }

  (*  Appliquons HNecDist à HNg pour obtenir N(exists v:U, v = x) *)
  apply HNecDist. exact HNg.
- (*  2. x n'est pas fini *)
```

```
    (*  Par A14, existence nécessaire implique non-finitude *)
    (*  A14 : N(exists y:U, y = x) <-> ~F_1 x *)
    apply A14.
    (*  Utilisons le résultat de la première partie *)
    (*  Nous avons déjà construit la cha\u00eene d'arguments ci-dessus *)
    assert (HNecImpl1: N((∃ v:U, v = g) → (∃ v:U, v = x))).
    { apply R7. exact HNec. }

    assert (HNecDist: N(∃ v:U, v = g) → N(∃ v:U, v = x)).
    { apply R6. exact HNecImpl1. }

    apply HNecDist. exact HNg.
Qed.
(*  P22: Tout ce qui suit d'un attribut de Dieu en tant que modifié est éternel et infini *)

(*  Prémisses: DP6, P14, P14-A, D1, P19, D8, A3, A14, R1, R6, R7 *)
Theorem P22 : ∀ x:U,
  (∃ g y y':U, G_1 g ∧ A_2 y g ∧ M_1 y' ∧ ¬F_1 y' ∧ N(∃ v:U, v = y') ∧
  K_2 y x ∧ K_2 y' x ∧ ~(∃ z:U, z ≠ y ∧ z ≠ y' ∧ K_2 z x)) →
  (N(∃ v:U, v = x) ∧ ¬F_1 x).
Proof.
  intros x H.
  destruct H as [g [y [y' [HGg [HA2yg [HM1y' [HNFy' [HNy' [HKyx [HKy'x HNoOtherCause]]]]]]]]]].
  (*  Par D6, si g est Dieu, alors g est une substance *)
  assert (HSg: S_1 g).
  { apply D6 in HGg. destruct HGg. exact H. }

  (*  Par DP7, si y est un attribut de g et g est une substance, alors y = g *)
  assert (Hyg: y = g).
  { apply DP7. split; assumption. }

  (*  Remplaçons y par g dans HKyx *)
  rewrite Hyg in HKyx.
  (*  Nous avons maintenant: HKyx : K_2 g x et HKy'x : K_2 y' x *)
  (*  La stratégie est de montrer que y' est un mode infini et nécessaire, et qu'il
     existe nécessairement. De ce fait, par A3, x existe aussi nécessairement. *)
  (*  Par A3, d'une cause déterminée (y') suit nécessairement un effet (x) *)
  assert (HNec_y'_x: N((∃ v:U, v = y') ↔ ∃ v:U, v = x)).
  { apply A3. exact HKy'x. }

  (*  Démontrons maintenant les deux parties du résultat *)
  split.
  - (*  1. x existe nécessairement *)
    (*  Utilisons HNec_y'_x pour extraire la partie gauche de la double implication *)
    assert (HNecImpl1: N((∃ v:U, v = y') → (∃ v:U, v = x))).
    {
      (*  Utilisons R7 pour extraire l'implication à partir de l'équivalence *)
      apply R7. exact HNec_y'_x.
    }
```

```
    (*  Appliquons R6 pour distribuer N sur l'implication *)
    (*  R6: N(P -> Q) -> (N(P) -> N(Q)) *)
    assert (HNecDist: N(∃ v:U, v = y') → N(∃ v:U, v = x)).
    { apply R6. exact HNecImpl1. }

    (*  Appliquons HNecDist à HNy' pour obtenir N(exists v:U, v = x) *)
    apply HNecDist. exact HNy'.
  - (*  2. x n'est pas fini *)
    (*  Par A14, existence nécessaire implique non-finitude *)
    (*  A14 : N(exists y:U, y = x) <-> ~F_1 x *)
    apply A14.
    (*  Utilisons le résultat de la première partie *)
    assert (HNecImpl1: N((∃ v:U, v = y') → (∃ v:U, v = x))).
    { apply R7. exact HNec_y'_x. }

    assert (HNecDist: N(∃ v:U, v = y') → N(∃ v:U, v = x)).
    { apply R6. exact HNecImpl1. }

    apply HNecDist. exact HNy'.
Qed.
(*  P23: Tout mode qui existe nécessairement et infiniment découle d'un attribut de Dieu *)

(*  Prémisses: P14, A9, P19, D8 *)
Theorem P23 : ∀ x:U,
  N(∃ v:U, v = x) →
  (∃ g y:U, G_1 g ∧ A_2 y g ∧ N((∃ v:U, v = y) → (∃ v:U, v = x))).
Proof.
  intros x HNx.
  (*  Par P14, il existe un Dieu unique g *)
  destruct P14 as [g [HGg HGodUnique]].
  (*  Par A9, tout objet a au moins un attribut *)
  assert (HAttrG: ∃ y:U, A_2 y g).
  { apply A9. }
  destruct HAttrG as [y HA2yg].
  (*  Notre témoin est (g, y) *)
  ∃ g, y.
  (*  Montrons trois choses:
     1. g est Dieu
     2. y est un attribut de g
     3. y est lié causalement à x *)
  (*  Première partie: g est Dieu *)
  split. { exact HGg. }

  (*  Deuxième partie: y est un attribut de g *)
  split. { exact HA2yg. }

  (*  Troisième partie: Montrer que y est causalement lié à x *)

  (*  Puisque g est Dieu, c'est une substance *)
```

```
  assert (HSg: S_1 g).
  { apply D6 in HGg. destruct HGg. exact H. }

  (*  Par DP7, si y est un attribut de g et g est une substance, alors y = g *)
  assert (Hyg: y = g).
  { apply DP7. split; [exact HA2yg | exact HSg]. }

  (*  Par P15, tout ce qui existe est en Dieu et conçu par Dieu *)
  assert (HIxg: I_2 x g ∧ C_2 x g).
  {
    pose proof (P15 x) as [h [HGh [HIxh HCxh]]].
    (*  g = h car ils sont tous deux Dieu, et Dieu est unique par P14_A *)
    assert (Hgh: g = h).
    {
      destruct P14_A as [k HP14A].
      assert (Hgk: g = k). { apply HP14A. exact HGg. }
      assert (Hhk: h = k). { apply HP14A. exact HGh. }
      rewrite Hgk. symmetry. exact Hhk.
    }

    (*  Remplaçons h par g *)
    rewrite ← Hgh in HIxh.
    rewrite ← Hgh in HCxh.
    split; assumption.
  }

  (*  Par A4, si x est conçu par g, alors g est cause de x *)
  assert (HKgx: K_2 g x).
  { apply A4. destruct HIxg as [_ HCxg]. exact HCxg. }

  (*  Puisque y = g, nous avons aussi K_2 y x *)
  assert (HKyx: K_2 y x).
  { rewrite Hyg. exact HKgx. }

  (*  Par A3, d'une cause déterminée suit nécessairement un effet *)
  assert (HNyx: N((∃ v:U, v = y) ↔ ∃ v:U, v = x)).
  { apply A3. exact HKyx. }

  (*  De cette équivalence nécessaire, nous pouvons extraire l'implication avec R7 *)
  assert (HNyimpx: N((∃ v:U, v = y) → (∃ v:U, v = x))).
  { apply R7. exact HNyx. }

  (*  C'est exactement ce que nous voulions montrer *)
  exact HNyimpx.
Qed.

(*  P24: L'essence des choses produites par Dieu n'implique pas l'existence *)
(*  Prémisses: D1 *)
Theorem P24 : ∀ x:U,
  (∃ g:U, G_1 g ∧ x ≠ g ∧ K_2 g x) → ¬L(∃ v:U, v = x).
Proof.
  intros x H.
```

```
  destruct H as [g [HGg [Hneq HKgx]]].
  (* Preuve par contradiction *)
  intro HLx.
  (* Si l'essence de x implique son existence (HLx), alors par D1,
     x est cause de soi-même et n'a pas de cause externe *)
  assert (HCausaSui: K_2 x x ∧ ¬ ∃ y:U , y ≠ x ∧ K_2 y x).
  { apply D1. exact HLx. }

  (* Mais nous savons que g est une cause de x et g \u2260 x *)
  destruct HCausaSui as [_ HNoExternalCause].
  (* Cela contredit directement le fait que g est une cause externe de x *)
  apply HNoExternalCause.
  ∃ g.
  split.
  - (* Nous devons montrer g <> x, mais nous avons x <> g *)
    (* Utilisons la symétrie de l'inégalité *)
    apply neq_sym. exact Hneq.
  - exact HKgx.
Qed.
(* P25: Dieu est cause efficiente de l'essence et de l'existence des choses *)
(* Prémisses: P15, A4 *)
Theorem P25 : ∀ x:U,
  ∃ g:U , G_1 g ∧ K_2 g x.
Proof.
  intro x.
  (* Par P15, tout ce qui existe est en Dieu et conçu par Dieu *)
  destruct (P15 x) as [g [HGg [HIxg HCxg]]].
  (* Le témoin est g, le Dieu dont nous venons d'établir l'existence *)
  ∃ g.
  (* Montrons deux choses:
     1. g est Dieu
     2. g est cause de x *)
  (* Première partie: g est Dieu *)
  split. { exact HGg. }

  (* Deuxième partie: g est cause de x *)
  (* Par A4, si x est conçu par g, alors g est cause de x *)
  apply A4. exact HCxg.
Qed.
(* P26: Une chose déterminée à produire un effet a été déterminée par Dieu *)
(* Prémisses: P16 *)
Theorem P26 : ∀ x y:U,
  (∃ z z':U , M_2 y z ∧ M_2 z' z ∧ K_2 x y) →
  (∃ g:U , G_1 g ∧ K_2 g y).
Proof.
  intros x y H.
  (* P26 est une conséquence directe de P16 *)
  (* P16 établit que Dieu est cause de toutes choses *)
  (* Donc il existe un Dieu g qui est cause de y *)
```

```
  (*  Appliquons P16 directement à y *)
  apply P16.
Qed.

(*  P27: Une chose déterminée par Dieu ne peut se rendre indéterminée *)
(*  Prémisses: P14-A, A3 *)
Theorem P27 : ∀ x:U,
  (∃ g:U, G_1 g ∧ K_2 g x ∧ ~(∃ z:U, z ≠ g ∧ K_2 z x)) →
  N(∃ v:U, v = x).
Proof.
  intros x H.
  destruct H as [g [HGg [HKgx HNoOtherCause]]].

  (*  Par P14-A, il existe un Dieu unique *)
  destruct P14_A as [h HP14A].

  (*  g = h car g est Dieu et h est le Dieu unique *)
  assert (Hgh: g = h).
  { apply HP14A. exact HGg. }

  (*  Par P19, Dieu est éternel *)
  assert (HEg: E_1 g).
  {
    destruct P19 as [k [HGk [HEk _]]].
    (*  g = k car g et k sont tous deux Dieu, et Dieu est unique *)
    assert (Hgk: g = k).
    {
      (*  Par HP14A, y est Dieu si et seulement si y = h *)
      (*  On a déjà prouvé que g = h *)
      assert (Hkh: k = h). { apply HP14A. exact HGk. }

      (*  Par transitivité, g = k *)
      rewrite Hgh. symmetry. exact Hkh.
    }
    rewrite Hgk. exact HEk.
  }

  (*  Par D8, l'éternité est l'existence nécessaire (logique) *)
  assert (HLg: L(∃ v:U, v = g)).
  { apply D8. exact HEg. }

  (*  Par R1, la nécessité logique implique la nécessité naturelle *)
  assert (HNg: N(∃ v:U, v = g)).
  { apply R1. exact HLg. }

  (*  Par A3, d'une cause déterminée suit nécessairement un effet *)
  assert (HNgx: N((∃ v:U, v = g) ↔ ∃ v:U, v = x)).
  { apply A3. exact HKgx. }

  (*  Utilisons R7 pour extraire l'implication de l'équivalence *)
  assert (HNgTox: N((∃ v:U, v = g) → (∃ v:U, v = x))).
  { apply R7. exact HNgx. }

  (*  Utilisons R6 pour distribuer N sur l'implication *)
```

```
  assert (HNDist: N(∃ v:U, v = g) → N(∃ v:U, v = x)).
  { apply R6. exact HNgTox. }

  (* Appliquons HNDist à HNg pour obtenir la nécessité de x *)
  apply HNDist. exact HNg.
Qed.

(* P28: Tout mode fini est déterminé à exister par un autre mode fini *)
(* Prémisses: P14-A, P16, A8, A4, A14, A3, R10, DP4, P8 *)
Theorem P28 : ∀ x:U,
  (F_1 x ∧ ¬N(∃ v:U, v = x)) →
  (∃ g:U, G_1 g ∧ K_2 g x ∧ (∀ y:U, I_2 x y → K_2 y x) ∧
   (∃ z:U, z ≠ x ∧ K_2 z x ∧ ¬N(∃ v:U, v = z) ∧ F_1 z)).
Proof.
  intros x [HFx HNotNx].

  (* Par P16, pour tout objet, il existe un Dieu qui en est la cause *)
  destruct (P16 x) as [g [HGg HKgx]].

  (* g est notre premier témoin *)
  ∃ g.

  (* 1. g est Dieu *)
  split. { exact HGg. }

  (* 2. g est cause de x *)
  split. { exact HKgx. }

  (* 3. Tout ce en quoi x est, est cause de x *)
  split.
  {
    intros y HIxy.
    (* Par A8, si x est en y, alors x est conçu par y *)
    assert (HCxy: C_2 x y). { apply A8. exact HIxy. }
    (* Par A4, si x est conçu par y, alors y est cause de x *)
    apply A4. exact HCxy.
  }

  (* 4. Il existe un mode fini z qui est cause de x *)

  (* Montrons d'abord que x est un mode (et non une substance) *)
  assert (HMx: M_1 x).
  {
    destruct (DP5 x) as [HSx | HMx].
    - (* Si x est une substance, alors x est infini (P8) *)
      assert (HNotFx: ¬F_1 x). { apply P8. exact HSx. }
      (* Contradiction avec notre hypothèse que x est fini *)
      contradiction.
    - exact HMx.
  }

  (* Par R10, si x est un mode fini non nécessaire, alors il existe un autre mode fini
     non nécessaire z qui est cause de x *)
  apply R10. split; assumption.
```

```
Qed.

(* P29: Rien n'est contingent dans la nature *)
(* Prémisses: P14-A, P16, P11, D7b, D8, D1 *)
Theorem P29 : ∃ g:U,
  G_1 g ∧ L(∃ x:U, x = g) ∧ (∀ x:U, x ≠ g → N_1 x).
Proof.
  (* Par P14-A, il existe un Dieu unique *)
  destruct P14_A as [g HP14A].

  (* g est notre témoin *)
  ∃ g.

  (* Nous allons montrer trois choses:
     1. g est Dieu
     2. g existe nécessairement
     3. Tout ce qui n'est pas Dieu est nécessaire au sens de D7b *)

  (* Première partie: g est Dieu *)
  assert (HGg: G_1 g).
  { apply HP14A. reflexivity. }

  split. { exact HGg. }

  (* Deuxième partie: g existe nécessairement *)
  split.
  {
    (* Par P11, Dieu existe nécessairement: L(exists x:U, G_1 x) *)
    (* Mais nous avons besoin de L(exists x:U, x = g) *)

    (* D'abord, montrons que (exists x:U, G_1 x) implique (exists x:U, x = g) *)
    assert (H_imp: (∃ x:U, G_1 x) → (∃ x:U, x = g)).
    {
      intro H_exists_god.
      destruct H_exists_god as [h HGh].

      (* Par P14-A, h = g car h est Dieu et g est le seul Dieu *)
      assert (Hhg: h = g).
      { apply HP14A. exact HGh. }

      (* Donc g existe *)
      ∃ g.
      reflexivity.
    }

    (* Par R5, on peut transformer cette implication en nécessité logique *)
    assert (H_nec_imp: L((∃ x:U, G_1 x) → (∃ x:U, x = g))).
    { apply R5. exact H_imp. }

    (* Par R3, on distribue la nécessité logique sur l'implication *)
    assert (H_dist: L(∃ x:U, G_1 x) → L(∃ x:U, x = g)).
    { apply R3. exact H_nec_imp. }

    (* Finalement, on applique cette implication à P11 *)
    apply H_dist.
```

```
    exact P11.
  }

  (*  Troisième partie: Tout ce qui n'est pas Dieu est nécessaire au sens de D7b *)
  intros x Hx_neq_g.
  (*  Par D7b, une chose est nécessaire quand elle est déterminée par autre chose *)
  apply D7b.
  (*  Par P16, pour toute chose, il existe un Dieu qui en est la cause *)
  destruct (P16 x) as [h [HGh HKhx]].
  (*  Par P14-A, h = g car h est Dieu et g est le seul Dieu *)
  assert (Hhg: h = g).
  { apply HP14A. exact HGh. }

  (*  Substituons h par g dans K_2 h x *)
  rewrite Hhg in HKhx.
  (*  Nous avons maintenant montré que K_2 g x *)
  ∃ g.
  split.
  - (*  g \u2260 x *)
    apply neq_sym. exact Hx_neq_g.
  - (*  g est cause de x *)
    exact HKhx.
Qed.
(*  P30: L'entendement doit comprendre les attributs et affections de Dieu *)
(*  Prémisses: DP5, A6, A9, D4b, D5a, D5b *)
Theorem P30 : ∀ x y:U,
  (A_1 x ∧ T_1 x ∧ O_2 y x) → (A_1 y ∨ M_1 y).
Proof.
  (*  Introduction des hypothèses *)
  intros x y [HA1x [HTx HOyx]].
  (*  Par DP5, y est soit une substance, soit un mode *)
  destruct (DP5 y) as [HSy | HMy].
  (*  Cas 1: y est une substance *)
  {
    (*  Si y est une substance, montrons que y est un attribut *)
    left.
    (*  Par A9, toute chose a au moins un attribut *)
    destruct (A9 y) as [z HA2zy].
    (*  Par D4b, z est un attribut et y est conçu à travers z *)
    apply D4b in HA2zy as [HA1z HCyz].
    (*  Par D3, une substance est en soi et conçue par soi *)
    apply D3 in HSy as [HIyy HCyy].
    (*  Par A2, si y est conçu par soi, il ne peut être conçu par autre chose *)
    apply A2 in HCyy as HNotCyz.
    (*  Donc y ne peut être conçu que par lui-même, ce qui signifie que z = y *)
    assert (Hzy: z = y).
    {
      (*  Preuve par contradiction *)
```

```
      apply NNPP. (*  Not Not P -> P *)
      intro Hneq.
      (*  Si z \u2260 y, alors y serait conçu à travers quelque chose d'autre que soi *)
      apply HNotCyz.
      ∃ z.
      split; assumption.
    }

    (*  Substituons z par y dans HA1z *)
    rewrite Hzy in HA1z.
    exact HA1z.
  }

  (*  Cas 2: y est un mode *)
  {
    (*  Si y est un mode, nous avons directement la conclusion *)
    right.
    exact HMy.
  }
Qed.
(*  P31a: L'entendement est un mode *)
(*  Prémisses: DP5, A17a, DPI *)
Theorem P31a : ∀ x:U,
  U_1 x → M_1 x.
Proof.
  (*  Introduction des hypothèses *)
  intros x HUx.
  (*  Par DP5, tout objet est soit une substance, soit un mode *)
  destruct (DPI x) as [[HSx HNotMx] | [HNotSx HMx]].
  (*  Cas 1: x est une substance *)
  {
    (*  Si x est une substance, alors par DPII, x est son propre attribut *)
    assert (HA2xx: A_2 x x).
    { apply DPII. exact HSx. }

    (*  Par D4b, A_2 x x implique que x est un attribut *)
    apply D4b in HA2xx as [HA1x _].
    (*  Par A17a, si x est un entendement, alors x n'est pas un attribut *)
    assert (HNotA1x: ¬A_1 x).
    { apply A17a. exact HUx. }

    (*  Contradiction: x est un attribut (HA1x) et x n'est pas un attribut (HNotA1x) *)
    contradiction.
  }

  (*  Cas 2: x est un mode *)
  {
    (*  Si x est un mode, c'est exactement ce que nous voulions démontrer *)
    exact HMx.
  }
```

```
Qed.

(*  P31b: La volonté est un mode *)
(*  Prémisses: DP5, A17b, DPI, DPII *)
Theorem P31b : ∀ x:U,
  W_1 x → M_1 x.
Proof.
  (*  Introduction des hypothèses *)
  intros x HWx.

  (*  Par DPI, tout objet est soit une substance, soit un mode *)
  destruct (DPI x) as [[HSx HNotMx] | [HNotSx HMx]].

  (*  Cas 1: x est une substance *)
  {
    (*  Si x est une substance, alors par DPII, x est son propre attribut *)
    assert (HA2xx: A_2 x x).
    { apply DPII. exact HSx. }

    (*  Par D4b, A_2 x x implique que x est un attribut *)
    apply D4b in HA2xx as [HA1x _].

    (*  Par A17b, si x est une volonté, alors x n'est pas un attribut *)
    assert (HNotA1x: ¬A_1 x).
    { apply A17b. exact HWx. }

    (*  Contradiction: x est un attribut (HA1x) et x n'est pas un attribut (HNotA1x) *)
    contradiction.
  }

  (*  Cas 2: x est un mode *)
  {
    (*  Si x est un mode, c'est exactement ce que nous voulions démontrer *)
    exact HMx.
  }
Qed.

(*  P31c: Le désir est un mode *)
(*  Prémisses: DPI, DPII, D4b, A17c *)
Theorem P31c : ∀ x:U,
  D_1 x → M_1 x.
Proof.
  (*  Introduction des hypothèses *)
  intros x HDx.

  (*  Par DPI, tout objet est soit une substance, soit un mode *)
  destruct (DPI x) as [[HSx HNotMx] | [HNotSx HMx]].

  (*  Cas 1: x est une substance *)
  {
    (*  Si x est une substance, alors par DPII, x est son propre attribut *)
    assert (HA2xx: A_2 x x).
    { apply DPII. exact HSx. }

    (*  Par D4b, A_2 x x implique que x est un attribut *)
    apply D4b in HA2xx as [HA1x _].
```

```
    (* Par A17c, si x est un désir, alors x n'est pas un attribut *)
    assert (HNotA1x: ¬A_1 x).
    { apply A17c. exact HDx. }

    (* Contradiction: x est un attribut (HA1x) et x n'est pas un attribut (HNotA1x) *)
    contradiction.
  }

  (* Cas 2: x est un mode *)
  {
    (* Si x est un mode, c'est exactement ce que nous voulions démontrer *)
    exact HMx.
  }
Qed.

(* P31d: L'amour est un mode *)
(* Prémisses: DPI, DPII, D4b, A17d *)
Theorem P31d : ∀ x:U,
  J_1 x → M_1 x.
Proof.
  (* Introduction des hypothèses *)
  intros x HJx.

  (* Par DPI, tout objet est soit une substance, soit un mode *)
  destruct (DPI x) as [[HSx HNotMx] | [HNotSx HMx]].

  (* Cas 1: x est une substance *)
  {
    (* Si x est une substance, alors par DPII, x est son propre attribut *)
    assert (HA2xx: A_2 x x).
    { apply DPII. exact HSx. }

    (* Par D4b, A_2 x x implique que x est un attribut *)
    apply D4b in HA2xx as [HA1x _].

    (* Par A17d, si x est une instance d'amour, alors x n'est pas un attribut *)
    assert (HNotA1x: ¬A_1 x).
    { apply A17d. exact HJx. }

    (* Contradiction: x est un attribut (HA1x) et x n'est pas un attribut (HNotA1x) *)
    contradiction.
  }

  (* Cas 2: x est un mode *)
  {
    (* Si x est un mode, c'est exactement ce que nous voulions démontrer *)
    exact HMx.
  }
Qed.

(* P32: La volonté ne peut être appelée cause libre *)
(* Prémisses: P31b, P16, D7a, D7b, DPI, D6 *)
Theorem P32 : ∀ x:U,
  W_1 x → (¬B_1 x ∧ N_1 x).
Proof.
```

```
(*  Introduction des hypothèses *)
intros x HWx.
(*  Par P31b, si x est une volonté, alors x est un mode *)
assert (HMx: M_1 x).
{ apply P31b. exact HWx. }

(*  Nous allons montrer deux choses:
   1. x n'est pas libre (~B_1 x)
   2. x est nécessaire (N_1 x) *)
split.
(*  1. La volonté n'est pas libre *)
{
  (*  Preuve par contradiction *)
  intro HBx.
  (*  Par D7a, si x est libre, alors x est cause de soi-même et n'a pas de cause externe *)

  apply D7a in HBx as [HKxx HNoExternalCause].
  (*  Par P16, pour tout objet x, il existe un Dieu g qui est cause de x *)
  destruct (P16 x) as [g [HGg HKgx]].
  (*  Par D6, g est une substance *)
  assert (HSg: S_1 g).
  { apply D6 in HGg. destruct HGg. exact H. }

  (*  g ne peut pas être égal à x *)
  assert (Hgx: g ≠ x).
  {
    intro Heq.
    (*  Si g = x, alors par substitution, x serait une substance *)
    assert (HSx: S_1 x).
    { rewrite ← Heq. exact HSg. }

    (*  Par DPI, une chose est soit une substance, soit un mode, mais pas les deux *)
    assert (H: (S_1 x ∧ ¬M_1 x) ∨ (¬S_1 x ∧ M_1 x)).
    { apply DPI. }

    (*  Or, nous savons que x est un mode (HMx) et une substance (HSx), contradiction *)

    destruct H as [[_ HNotMx] | [HNotSx _]].
    - apply HNotMx. exact HMx.
    - apply HNotSx. exact HSx.
  }

  (*  Puisque g est cause de x (HKgx) et g \u2260 x, x a une cause externe *)
  assert (HExternalCause: ∃ z:U , z ≠ x ∧ K_2 z x).
  {
    ∃ g. split.
    - exact Hgx.
    - exact HKgx.
  }
```

```
    (*  Contradiction avec HNoExternalCause *)
    apply HNoExternalCause. exact HExternalCause.
  }

  (*  2. La volonté est nécessaire *)
  {
    (*  Par D7b, x est nécessaire ssi x est déterminé par une cause externe *)
    apply D7b.
    (*  Par P16, pour tout objet x, il existe un Dieu g qui est cause de x *)
    destruct (P16 x) as [g [HGg HKgx]].
    (*  Par D6, g est une substance *)
    assert (HSg: S_1 g).
    { apply D6 in HGg. destruct HGg. exact H. }

    (*  g ne peut pas être égal à x (même preuve que dans la première partie) *)
    assert (Hgx: g ≠ x).
    {
      intro Heq.
      (*  Si g = x, alors par substitution, x serait une substance *)
      assert (HSx: S_1 x).
      { rewrite ← Heq. exact HSg. }

      (*  Par DPI, une chose est soit une substance, soit un mode, mais pas les deux *)
      assert (H: (S_1 x ∧ ¬M_1 x) ∨ (¬S_1 x ∧ M_1 x)).
      { apply DPI. }

      (*  Or, nous savons que x est un mode (HMx) et une substance (HSx), contradiction *)

      destruct H as [[_ HNotMx] | [HNotSx _]].
      - apply HNotMx. exact HMx.
      - apply HNotSx. exact HSx.
    }

    (*  g est notre témoin pour la cause externe de x *)
    ∃ g.
    split; [exact Hgx | exact HKgx].
  }
Qed.
(*  P33: Les choses n'auraient pu être produites par Dieu d'aucune autre manière *)
(*  Note: Jarrett indique que cette proposition n'est pas dérivable de son système formel *)

(*  Pour la démontrer, nous ajoutons un axiome supplémentaire R11 qui formalise le nécessitarisme divin

Axiom R11 : ∀ g y z:U,
  G_1 g → K_2 g y → K_2 g z → y = z ∨ ~(∃ v:U, v = y) ∨ ~(∃ v:U, v = z).
(*  Axiome R12: Si une chose existe nécessairement, elle existe dans tous les mondes possibles *)

Axiom R12 : ∀ P:Prop,
  N(P) → ∀ Q:Prop, M(Q) → M(P ∧ Q).
Theorem P33 : ∀ x:U,
```

```
  ∃ g:U, G_1 g →
  L(∀ y:U, K_2 g y → ¬M(∃ z:U, z ≠ y ∧ K_2 g z)).
Proof.
  intro x.
  (* Par P14, il existe un Dieu unique g *)
  destruct P14 as [g [HGg HUnique]].
  ∃ g.
  (* Introduction des hypothèses *)
  intro HGg'.
  (* Par les axiomes modaux, nous pouvons établir la nécessité de cette proposition *)
  apply R5. (* Règle de nécessitation: P -> L(P) *)
  (* Nous devons prouver: forall y:U, K_2 g y -> ~M(exists z:U, z <> y /\ K_2 g z) *)
  intros y HKgy.
  (* Preuve par contradiction *)
  intro HMz.
  (* Si M(exists z:U, z <> y /\ K_2 g z), alors dans un certain monde possible,
     il existe un z tel que z \u2260 y et Dieu cause z *)
  (* Par P27, si une chose est déterminée par Dieu, elle existe nécessairement *)
  assert (HNy: N(∃ v:U, v = y)).
  {
    (* Pour appliquer P27, nous devons montrer que g est l'unique cause de y *)
    apply P27.
    ∃ g.
    (* Montrons trois choses:
       1. g est Dieu
       2. g est cause de y
       3. g est la seule cause de y *)
    split. { exact HGg. }
    split. { exact HKgy. }

    (* Montrons que seul g est cause première de y *)
    intro HExistsCause.
    destruct HExistsCause as [w [Hwg HKwy]].
    (* Par P14, il n'existe qu'une seule substance, qui est Dieu *)
    assert (HSw_or_HMw: S_1 w ∨ M_1 w).
    { apply DP5. }

    destruct HSw_or_HMw as [HSw | HMw].
    - (* Si w est une substance, alors w = g car g est la seule substance (par P14) *)
      assert (Hwg': w = g).
      { apply HUnique. exact HSw. }
      contradiction.
    - (* Si w est un mode, alors w n'est pas une cause première *)
      (* REMARQUE: Nous admettons ce fait comme un axiome implicite du système spinoziste *)

      admit.
  }
```

```
  (* Si y existe nécessairement (HNy), alors par R9, y existe dans tous les mondes possibles *)

  assert (HNotMNoty: ¬M(~(∃ v:U, v = y))).
  { apply R9. exact HNy. }

  (* Par R12, si y existe nécessairement et qu'il est possible que z existe (o\u00f9 z \u2260 y et g ca
     alors il est possible que y et z existent ensemble, avec g causant les deux *)
  assert (HM_y_and_z: M((∃ v:U, v = y) ∧ (∃ z:U, z ≠ y ∧ K_2 g z))).
  { apply R12. exact HNy. exact HMz. }

  (* De HM_y_and_z, nous déduisons qu'il existe un monde possible o\u00f9:
     1. y existe
     2. Il existe un z tel que z \u2260 y et g cause z
     3. g cause également y (de notre hypothèse HKgy) *)

  (* Dans ce monde, par l'axiome R11, cela est impossible car:
     - y \u2260 z
     - y existe
     - z existe
     - Les deux sont causés par g *)

  (* Pour compléter formellement notre contradiction, nous utilisons un argument sémantique:
     la possibilité établie par HM_y_and_z contredit l'axiome R11 du système.

     Nous terminons donc en Admitted pour accepter cette partie du raisonnement qui
     nécessiterait une formalisation spécifique de la sémantique des mondes possibles
     que nous n'avons pas dans notre système actuel. *)
Admitted.
(* P34: La puissance de Dieu est son essence même *)
(* Prémisses: DP7, P14-A, D6, D3, D4b, A19 *)
Theorem P34 : ∀ x:U,
  ∃ g:U, G_1 g ∧ (A_2 x g ↔ P_2 x g).
Proof.
  intro x.
  (* Par P14, il existe un Dieu unique g *)
  destruct P14 as [g [HGg HUnique]].
  ∃ g.
  (* Montrons deux choses:
     1. g est Dieu
     2. x est un attribut de g si et seulement si x est la puissance de g *)
  split.
  { exact HGg. }

  (* Maintenant, montrons l'équivalence A_2 x g <-> P_2 x g *)
  split.
  (* => Première direction: si x est un attribut de g, alors x est la puissance de g *)
  {
    intro HA2xg.
    (* Par D6, si g est Dieu, alors g est une substance *)
    assert (HSg: S_1 g).
```

```
{ apply D6 in HGg. destruct HGg. exact H. }

(* Par D4b, si x est un attribut de g, alors x est un attribut et g est conçu à travers x *)

apply D4b in HA2xg.
destruct HA2xg as [HA1x HCgx].
(* Par DP7, si x est un attribut de g et g est une substance, alors x = g *)
assert (Hxg: x = g).
{
  (* CORRECTION: Construction explicite de la conjonction *)
  apply DP7.
  split.
  - apply D4b. split; [exact HA1x | exact HCgx].
  - exact HSg.
}

(* Par D3, si g est une substance, alors g est en soi et conçu par soi *)
assert (H_in_and_conceived: I_2 g g ∧ C_2 g g).
{ apply D3. exact HSg. }

(* De Hxg, nous avons x = g, donc I_2 x g <-> I_2 g g et C_2 x g <-> C_2 g g *)

(* Par A19, si x est à la fois en y, conçu par y, et vice versa, alors x est la puissance de y *)

(* A19: (((I_2 x y /\ C_2 x y) /\ I_2 y x) /\ C_2 y x) = P_2 x y *)

(* Remplaçons chaque occurrence de x par g dans l'équation *)
assert (HP2xg: P_2 x g).
{
  (* Puisque x = g, nous avons:
     - I_2 x g = I_2 g g (g est en soi)
     - C_2 x g = C_2 g g (g est conçu par soi)
     - I_2 g x = I_2 g g (g est en soi)
     - C_2 g x = C_2 g g (g est conçu par soi) *)

  (* Réécrivons l'équation de A19 en substituant x par g *)

  (* Nous réécrivons x = g *)
  rewrite Hxg.
  (* Par A19, nous avons le résultat direct *)
  destruct H_in_and_conceived as [HIgg HCgg].
  rewrite ← A19.
  repeat split; assumption.
}

(* Donc x est la puissance de g *)
exact HP2xg.
}

(* <= Seconde direction: si x est la puissance de g, alors x est un attribut de g *)
{
```

```
    intro HP2xg.
    (*  Par A19, si x est la puissance de g, alors:
       (((I_2 x g /\ C_2 x g) /\ I_2 g x) /\ C_2 g x) *)
    assert (H_relations: ((I_2 x g ∧ C_2 x g) ∧ I_2 g x) ∧ C_2 g x).
    {
      (*  A19 établit une égalité entre cette expression et P_2 x g *)
      pose proof (A19 x g) as Heq.
      (*  Puisque nous avons P_2 x g (HP2xg), nous pouvons utiliser l'égalité *)
      rewrite ← Heq in HP2xg.
      exact HP2xg.
    }

    (*  Décomposons cette assertion complexe *)
    destruct H_relations as [H_part1 HCgx].
    destruct H_part1 as [H_part2 HIgx].
    destruct H_part2 as [HIxg HCxg].
    (*  Par D6, si g est Dieu, alors g est une substance *)
    assert (HSg: S_1 g).
    { apply D6 in HGg. destruct HGg. exact H. }

    (*  Par D4b, x est un attribut de g ssi x est un attribut et g est conçu par x *)
    apply D4b. split.
    (*  1. Montrons que x est un attribut *)
    - apply D4a. ∃ g.
      (*  g est une substance *)
      split. { exact HSg. }
      (*  x est en g, x est conçu par g, g est en x, g est conçu par x *)
      repeat split; assumption.
    (*  2. g est conçu par x *)
    - exact HCgx.
  }
Qed.
(*  P35: Tout ce qui existe est nécessaire *)
(*  Prémisses: identiques à P29 *)
Theorem P35 : ∃ g:U,
  G_1 g ∧ L(∃ x:U, x = g) ∧ (∀ x:U, x ≠ g → N_1 x).
Proof.
  (*  Par P14-A, il existe un Dieu unique *)
  destruct P14_A as [g HP14A].
  (*  g est notre témoin *)
  ∃ g.
  (*  Nous allons montrer trois choses:
     1. g est Dieu
     2. g existe nécessairement
     3. Tout ce qui n'est pas Dieu est nécessaire au sens de D7b *)
  (*  Première partie: g est Dieu *)
  assert (HGg: G_1 g).
  { apply HP14A. reflexivity. }
```

```
split. { exact HGg. }

(*  Deuxième partie: g existe nécessairement *)
split.
{
  (*  Par P11, Dieu existe nécessairement: L(exists x:U, G_1 x) *)
  (*  Mais nous avons besoin de L(exists x:U, x = g) *)

  (*  D'abord, montrons que (exists x:U, G_1 x) implique (exists x:U, x = g) *)
  assert (H_imp: (∃ x:U, G_1 x) → (∃ x:U, x = g)).
  {
    intro H_exists_god.
    destruct H_exists_god as [h HGh].
    (*  Par P14-A, h = g car h est Dieu et g est le seul Dieu *)
    assert (Hhg: h = g).
    { apply HP14A. exact HGh. }

    (*  Donc g existe *)
    ∃ g.
    reflexivity.
  }

  (*  Par R5, on peut transformer cette implication en nécessité logique *)
  assert (H_nec_imp: L((∃ x:U, G_1 x) → (∃ x:U, x = g))).
  { apply R5. exact H_imp. }

  (*  Par R3, on distribue la nécessité logique sur l'implication *)
  assert (H_dist: L(∃ x:U, G_1 x) → L(∃ x:U, x = g)).
  { apply R3. exact H_nec_imp. }

  (*  Finalement, on applique cette implication à P11 *)
  apply H_dist.
  exact P11.
}

(*  Troisième partie: Tout ce qui n'est pas Dieu est nécessaire au sens de D7b *)
intros x Hx_neq_g.
(*  Par D7b, une chose est nécessaire quand elle est déterminée par autre chose *)
apply D7b.
(*  Par P16, pour toute chose, il existe un Dieu qui en est la cause *)
destruct (P16 x) as [h [HGh HKhx]].
(*  Par P14-A, h = g car h est Dieu et g est le seul Dieu *)
assert (Hhg: h = g).
{ apply HP14A. exact HGh. }

(*  Substituons h par g dans K_2 h x *)
rewrite Hhg in HKhx.
(*  Nous avons maintenant montré que K_2 g x *)
∃ g.
split.
```

```
  - (*  g \u2260 x *)
    apply neq_sym. exact Hx_neq_g.
  - (*  g est cause de x *)
    exact HKhx.
Qed.

(*  P36: Il n'existe rien dont la nature ne produise quelque effet *)
(*  Note: Jarrett indique que cette proposition n'est pas dérivable de son système formel *)

Theorem P36 : ∀ x:U,
  (∃ v:U, v = x) → (∃ y:U, K_2 x y).
Proof.
  (*  Introduction des hypothèses *)
  intros x Hexists.

  (*  Nous allons examiner si x est une substance ou un mode *)
  destruct (DP5 x) as [HSx | HMx].

  (*  Cas 1: x est une substance *)
  {
    (*  Par DPIII, toute substance est cause de soi-même *)
    assert (HKxx: K_2 x x).
    { apply DPIII. exact HSx. }

    (*  x est cause de soi-même, donc il existe un y (à savoir x) tel que K_2 x y *)
    ∃ x.
    exact HKxx.
  }

  (*  Cas 2: x est un mode *)
  {
    (*  Par D5b, si x est un mode, alors il existe une substance s dont x est un mode *)
    apply D5b in HMx.
    destruct HMx as [s [HSs HM_x_s]].

    (*  Par le lemme DP4, toute substance est cause de soi-même *)
    assert (HKss: K_2 s s).
    { apply DP4. exact HSs. }

    (*  Par D5a, si x est un mode de s, alors x est en s et conçu par s *)
    apply D5a in HM_x_s.
    destruct HM_x_s as [Hneq [HIxs HCxs]].

    (*  Par la définition de Spinoza, la nature d'une chose détermine ses effets.
       Si x est un mode, son essence est déterminée par la substance dont il est un mode.
       Nous ne pouvons pas directement prouver que x produit un effet sans hypothèses
       supplémentaires sur la nature des modes. *)

    (*  Cependant, nous pouvons invoquer P16 qui établit que Dieu est cause de toutes choses *)

    destruct (P16 x) as [g [HGg HKgx]].

    (*  Puisque g est Dieu, il a une infinité d'attributs et produit une infinité d'effets,
       dont l'un est x. Par transitivité de la causalité (qui n'est pas formellement
       établie dans le système de Jarrett), x doit également produire des effets. *)

    (*  Nous devons admettre ce point par la cohérence du système spinoziste, car comme
```

```
    l'indique Jarrett, P36 n'est pas directement dérivable dans son système formel. *)
  (*  Pour compléter la preuve en utilisant explicitement les axiomes disponibles,
    nous pouvons ajouter un axiome supplémentaire qui formalise cette propriété
    de la causalité dans le système spinoziste:

    Axiom Transitive_Causality: forall x y z:U, K_2 x y /\ K_2 y z -> exists w:U, K_2 y w.

    Mais puisque nous n'avons pas cet axiome, nous admettons cette étape. *)
  (*  Puisque le système de Spinoza est déterministe et que toute chose découle
    nécessairement de l'essence divine, nous pouvons affirmer que tout mode
    a un effet, même si nous ne pouvons pas le démontrer formellement. *)
  admit.
}
Admitted.
End SpinozaJarrett.
```