A Formalization Of The Dutch Book Argument

Matthew Doty

April 29, 2020

Contents

1 Minimal Logic

theory *Minimal-Logic*
  imports *Main*
begin

This theory presents *minimal logic*, the implicational fragment of intuitionistic logic.

1.1 Axiomatization

Minimal logic is given by the following Hilbert-style axiom system:

class *Minimal-Logic* =
  fixes *deduction* :: $'a \Rightarrow bool$ ($\vdash$ - [*60*] *55*)
  fixes *implication* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixr** $\rightarrow$ *70*)
  assumes *Axiom-1*: $\vdash \varphi \rightarrow \psi \rightarrow \varphi$
  assumes *Axiom-2*: $\vdash (\varphi \rightarrow \psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \chi$
  assumes *Modus-Ponens*: $\vdash \varphi \rightarrow \psi \Longrightarrow \vdash \varphi \Longrightarrow \vdash \psi$

A convenience class to have is *Minimal-Logic* extended with a single named constant, intended to be *falsum*. Other classes extending this class will provide rules for how this constant interacts with other terms.

class *Minimal-Logic-With-Falsum* = *Minimal-Logic* +
  fixes *falsum* :: $'a$ ($\bot$)

1.2 Common Rules

lemma (**in** *Minimal-Logic*) *trivial-implication*: $\vdash \varphi \rightarrow \varphi$
  by (*meson Axiom-1 Axiom-2 Modus-Ponens*)

lemma (**in** *Minimal-Logic*) *flip-implication*: $\vdash (\varphi \rightarrow \psi \rightarrow \chi) \rightarrow \psi \rightarrow \varphi \rightarrow \chi$
  by (*meson Axiom-1 Axiom-2 Modus-Ponens*)

lemma (**in** *Minimal-Logic*) *hypothetical-syllogism*: $\vdash (\psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \chi$
  by (*meson Axiom-1 Axiom-2 Modus-Ponens*)

lemma (**in** *Minimal-Logic*) *flip-hypothetical-syllogism*:
  shows $\vdash (\psi \rightarrow \varphi) \rightarrow (\varphi \rightarrow \chi) \rightarrow (\psi \rightarrow \chi)$
  using *Modus-Ponens flip-implication hypothetical-syllogism* **by** *blast*

lemma (**in** *Minimal-Logic*) *implication-absorption*: $\vdash (\varphi \rightarrow \varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \psi$
  by (*meson Axiom-1 Axiom-2 Modus-Ponens*)

1.3 Lists of Assumptions

1.3.1 List Implication

Implication given a list of assumptions can be expressed recursively

primrec (**in** *Minimal-Logic*) *list-implication* :: *'a list* ⇒ *'a* ⇒ *'a* (**infix** $:\to$ *80*)
where
$[] :\to \varphi = \varphi$
$| \ (\psi \ \# \ \Psi) :\to \varphi = \psi \to \Psi :\to \varphi$

1.3.2 Definition of Deduction

Deduction from a list of assumptions can be expressed in terms of $(:\to)$.

definition (**in** *Minimal-Logic*) *list-deduction* :: *'a list* ⇒ *'a* ⇒ *bool* (**infix** $:\vdash$ *60*)
where
$\Gamma :\vdash \varphi \equiv \vdash \Gamma :\to \varphi$

1.3.3 Interpretation as Minimal Logic

The relation $(:\vdash)$ may naturally be interpreted as a *proves* predicate for an instance of minimal logic for a fixed list of assumptions Γ.

Analogues of the two axioms of minimal logic can be naturally stated using list implication.

lemma (**in** *Minimal-Logic*) *list-implication-Axiom-1*: $\vdash \varphi \to \Gamma :\to \varphi$
by (*induct* Γ, (*simp*, *meson Axiom-1 Axiom-2 Modus-Ponens*)+)

lemma (**in** *Minimal-Logic*) *list-implication-Axiom-2*: $\vdash \Gamma :\to (\varphi \to \psi) \to \Gamma :\to \varphi \to \Gamma :\to \psi$
by (*induct* Γ, (*simp*, *meson Axiom-1 Axiom-2 Modus-Ponens hypothetical-syllogism*)+)

The lemmas $\vdash ?\varphi \to ?\Gamma :\to ?\varphi$ and $\vdash ?\Gamma :\to (?\varphi \to ?\psi) \to ?\Gamma :\to ?\varphi \to ?\Gamma :\to ?\psi$ jointly give rise to an interpretation of minimal logic, where a list of assumptions Γ plays the role of a *background theory* of $(:\vdash)$.

context *Minimal-Logic* **begin**
interpretation *List-Deduction-Logic*: *Minimal-Logic* $\lambda\ \varphi.\ \Gamma :\vdash \varphi$ $(\to)$
proof qed (*meson list-deduction-def*
Axiom-1
Axiom-2
Modus-Ponens
list-implication-Axiom-1
list-implication-Axiom-2)+
end

The following *weakening* rule can also be derived.

lemma (**in** *Minimal-Logic*) *list-deduction-weaken*: $\vdash \varphi \Longrightarrow \Gamma :\vdash \varphi$
unfolding *list-deduction-def*
using *Modus-Ponens list-implication-Axiom-1*
by *blast*

In the case of the empty list, the converse may be established.

lemma (**in** *Minimal-Logic*) *list-deduction-base-theory* [*simp*]: $[] :\vdash \varphi \equiv \vdash \varphi$

unfolding *list-deduction-def*
by *simp*

lemma (**in** *Minimal-Logic*) *list-deduction-modus-ponens*: $\Gamma :\vdash \varphi \rightarrow \psi \Longrightarrow \Gamma :\vdash \varphi \Longrightarrow \Gamma :\vdash \psi$
unfolding *list-deduction-def*
using *Modus-Ponens list-implication-Axiom-2*
by *blast*

1.4 The Deduction Theorem

One result in the meta-theory of minimal logic is the *deduction theorem*, which is a mechanism for moving antecedents back and forth from collections of assumptions.

To develop the deduction theorem, the following two lemmas generalize $\vdash (?\varphi \rightarrow ?\psi \rightarrow ?\chi) \rightarrow ?\psi \rightarrow ?\varphi \rightarrow ?\chi$.

lemma (**in** *Minimal-Logic*) *list-flip-implication1*: $\vdash (\varphi \# \Gamma) :\rightarrow \chi \rightarrow \Gamma :\rightarrow (\varphi \rightarrow \chi)$
by (*induct* Γ,
(*simp*, *meson Axiom-1 Axiom-2 Modus-Ponens flip-implication hypothetical-syllogism*)+)

lemma (**in** *Minimal-Logic*) *list-flip-implication2*: $\vdash \Gamma :\rightarrow (\varphi \rightarrow \chi) \rightarrow (\varphi \# \Gamma) :\rightarrow \chi$
by (*induct* Γ,
(*simp*, *meson Axiom-1 Axiom-2 Modus-Ponens flip-implication hypothetical-syllogism*)+)

Together the two lemmas above suffice to prove a form of the deduction theorem:

theorem (**in** *Minimal-Logic*) *list-deduction-theorem*: $(\varphi \# \Gamma) :\vdash \psi = \Gamma :\vdash \varphi \rightarrow \psi$
unfolding *list-deduction-def*
by (*metis Modus-Ponens list-flip-implication1 list-flip-implication2*)

1.5 Monotonic Growth in Deductive Power

In logic, for two sets of assumptions Φ and Ψ, if $\Psi \subseteq \Phi$ then the latter theory Φ is said to be *stronger* than former theory Ψ. In principle, anything a weaker theory can prove a stronger theory can prove. One way of saying this is that deductive power increases monotonically with as the set of underlying assumptions grow.

The monotonic growth of deductive power can be expressed as a meta-theorem in minimal logic.

The lemma $\vdash ?\Gamma :\rightarrow (?\varphi \rightarrow ?\chi) \rightarrow (?\varphi \# ?\Gamma) :\rightarrow ?\chi$ presents a means of *introducing* assumptions into a list of assumptions when those assumptions have arrived at an implication. The next lemma presents a means of *discharging* those assumptions, which can be used in the monotonic growth theorem to be proved.

lemma (**in** *Minimal-Logic*) *list-implication-removeAll*:
 $\vdash \Gamma :\to \psi \to (\mathit{removeAll}\ \varphi\ \Gamma) :\to (\varphi \to \psi)$
proof $-$
 have $\forall\ \psi.\ \vdash \Gamma :\to \psi \to (\mathit{removeAll}\ \varphi\ \Gamma) :\to (\varphi \to \psi)$
 proof(*induct* Γ)
  case *Nil*
  then show *?case* **by** (*simp*, *meson Axiom-1*)
 next
  case (*Cons* χ Γ)
  assume *inductive-hypothesis*: $\forall\ \psi.\ \vdash \Gamma :\to \psi \to \mathit{removeAll}\ \varphi\ \Gamma :\to (\varphi \to \psi)$
  moreover {
   assume $\varphi \neq \chi$
   with *inductive-hypothesis*
   have $\forall\ \psi.\ \vdash (\chi\ \#\ \Gamma) :\to \psi \to \mathit{removeAll}\ \varphi\ (\chi\ \#\ \Gamma) :\to (\varphi \to \psi)$
    by (*simp*, *meson Modus-Ponens hypothetical-syllogism*)
  }
  moreover {
   fix ψ
   assume *φ-equals-χ*: $\varphi = \chi$
   moreover with *inductive-hypothesis*
   have $\vdash \Gamma :\to (\chi \to \psi) \to \mathit{removeAll}\ \varphi\ (\chi\ \#\ \Gamma) :\to (\varphi \to \chi \to \psi)$ **by** *simp*
   hence $\vdash \Gamma :\to (\chi \to \psi) \to \mathit{removeAll}\ \varphi\ (\chi\ \#\ \Gamma) :\to (\varphi \to \psi)$
   by (*metis calculation Modus-Ponens implication-absorption list-flip-implication1*
     list-flip-implication2 list-implication.simps(*2*))
   ultimately have $\vdash (\chi\ \#\ \Gamma) :\to \psi \to \mathit{removeAll}\ \varphi\ (\chi\ \#\ \Gamma) :\to (\varphi \to \psi)$
    by (*simp*, *metis Modus-Ponens hypothetical-syllogism list-flip-implication1*
      list-implication.simps(*2*))
  }
  ultimately show *?case* **by** *simp*
 qed
 thus *?thesis* **by** *blast*
qed

From lemma above presents what is needed to prove that deductive power for lists is monotonic.

theorem (**in** *Minimal-Logic*) *list-implication-monotonic*:
 $\mathit{set}\ \Sigma \subseteq \mathit{set}\ \Gamma \Longrightarrow \vdash \Sigma :\to \varphi \to \Gamma :\to \varphi$
proof $-$
 assume $\mathit{set}\ \Sigma \subseteq \mathit{set}\ \Gamma$
 moreover have $\forall\ \Sigma\ \varphi.\ \mathit{set}\ \Sigma \subseteq \mathit{set}\ \Gamma \longrightarrow \vdash \Sigma :\to \varphi \to \Gamma :\to \varphi$
 proof(*induct* Γ)
  case *Nil*
  then show *?case*
  by (*metis list-implication.simps*(*1*) *list-implication-Axiom-1 set-empty subset-empty*)
 next
  case (*Cons* ψ Γ)
  assume *inductive-hypothesis*: $\forall \Sigma\ \varphi.\ \mathit{set}\ \Sigma \subseteq \mathit{set}\ \Gamma \longrightarrow \vdash \Sigma :\to \varphi \to \Gamma :\to \varphi$

```
  {
   fix Σ
   fix φ
   assume Σ-subset-relation: set Σ ⊆ set (ψ # Γ)
   have ⊢ Σ :→ φ → (ψ # Γ) :→ φ
   proof −
    {
     assume set Σ ⊆ set Γ
     hence ?thesis
       by (metis inductive-hypothesis Axiom-1 Modus-Ponens flip-implication
                list-implication.simps(2))
    }
    moreover {
     let ?Δ = removeAll ψ Σ
     assume ~ (set Σ ⊆ set Γ)
     hence set ?Δ ⊆ set Γ using Σ-subset-relation by auto
      hence ⊢ ?Δ :→ (ψ → φ) → Γ :→ (ψ → φ) using inductive-hypothesis
by auto
     hence ⊢ ?Δ :→ (ψ → φ) → (ψ # Γ) :→ φ
       by (metis Modus-Ponens
                flip-implication
                list-flip-implication2
                list-implication.simps(2))
     moreover have ⊢ Σ :→ φ → ?Δ :→ (ψ → φ)
       by (simp add: local.list-implication-removeAll)
     ultimately have ?thesis
       using Modus-Ponens hypothetical-syllogism by blast
    }
    ultimately show ?thesis by blast
   qed
  }
  thus ?case by simp
 qed
 ultimately show ?thesis by simp
qed
```

A direct consequence is that deduction from lists of assumptions is monotonic as well:

```
theorem (in Minimal-Logic) list-deduction-monotonic:
  set Σ ⊆ set Γ ⟹ Σ :⊢ φ ⟹ Γ :⊢ φ
  unfolding list-deduction-def
  using Modus-Ponens list-implication-monotonic
  by blast
```

1.6 The Deduction Theorem Revisited

The monotonic nature of deduction allows us to prove another form of the deduction theorem, where the assumption being discharged is completely removed from the list of assumptions.

theorem (**in** *Minimal-Logic*) *alternate-list-deduction-theorem*:
$(\varphi \# \Gamma) :\vdash \psi = (\mathit{removeAll}\ \varphi\ \Gamma) :\vdash \varphi \to \psi$
by (*metis list-deduction-def*
Modus-Ponens
filter-is-subset
list-deduction-monotonic
list-deduction-theorem
list-implication-removeAll
removeAll.simps(*2*)
removeAll-filter-not-eq)

1.7 Reflection

In logic the *reflection* principle sometimes refers to when a collection of assumptions can deduce any of its members. It is automatically derivable from $[\![\mathit{set}\ ?\Sigma \subseteq \mathit{set}\ ?\Gamma;\ ?\Sigma :\vdash\ ?\varphi]\!] \Longrightarrow\ ?\Gamma :\vdash\ ?\varphi$ among the other rules provided.

lemma (**in** *Minimal-Logic*) *list-deduction-reflection*: $\varphi \in \mathit{set}\ \Gamma \Longrightarrow \Gamma :\vdash \varphi$
by (*metis list-deduction-def*
insert-subset
list.simps(*15*)
list-deduction-monotonic
list-implication.simps(*2*)
list-implication-Axiom-1
order-refl)

1.8 The Cut Rule

Cut is a rule commonly presented in sequent calculi, dating back to Gerhard Gentzen's "Investigations in Logical Deduction" (1934) TODO: Cite me

The cut rule is not generally necessary in sequent calculi. It can often be shown that the rule can be eliminated without reducing the power of the underlying logic. However, as demonstrated by George Boolos' *Don't Eliminate Cute* (1984) (TODO: Cite me), removing the rule can often lead to very inefficient proof systems.

Here the rule is presented just as a meta theorem.

theorem (**in** *Minimal-Logic*) *list-deduction-cut-rule*:
$(\varphi \# \Gamma) :\vdash \psi \Longrightarrow \Delta :\vdash \varphi \Longrightarrow \Gamma\ @\ \Delta :\vdash \psi$
by (*metis*
(*no-types*, *lifting*)
Un-upper1
Un-upper2
list-deduction-modus-ponens
list-deduction-monotonic
list-deduction-theorem

```
    set-append)
```

The cut rule can also be strengthened to entire lists of propositions.

```
theorem (in Minimal-Logic) strong-list-deduction-cut-rule:
 (Φ @ Γ) :⊢ ψ ⟹ ∀ φ ∈ set Φ. Δ :⊢ φ ⟹ Γ @ Δ :⊢ ψ
proof −
  have ∀ ψ. (Φ @ Γ :⊢ ψ ⟶ (∀ φ ∈ set Φ. Δ :⊢ φ) ⟶ Γ @ Δ :⊢ ψ)
    proof(induct Φ)
      case Nil
      then show ?case
          by (metis Un-iff append.left-neutral list-deduction-monotonic set-append
subsetI)
    next
      case (Cons χ Φ)
      assume inductive-hypothesis:
        ∀ ψ. Φ @ Γ :⊢ ψ ⟶ (∀φ∈set Φ. Δ :⊢ φ) ⟶ Γ @ Δ :⊢ ψ
      {
        fix ψ χ
        assume (χ # Φ) @ Γ :⊢ ψ
        hence A: Φ @ Γ :⊢ χ → ψ  using list-deduction-theorem by auto
        assume ∀φ ∈ set (χ # Φ). Δ :⊢ φ
        hence B: ∀ φ ∈ set Φ. Δ :⊢ φ
          and C: Δ :⊢ χ by auto
        from A B have Γ @ Δ :⊢ χ → ψ using inductive-hypothesis by blast
        with C have Γ @ Δ :⊢ ψ
          by (meson list.set-intros(1)
                    list-deduction-cut-rule
                    list-deduction-modus-ponens
                    list-deduction-reflection)
      }
      thus ?case by simp
    qed
    moreover assume (Φ @ Γ) :⊢ ψ
  moreover assume ∀ φ ∈ set Φ. Δ :⊢ φ
  ultimately show ?thesis by blast
qed
```

1.9 Sets of Assumptions

While deduction in terms of lists of assumptions is straight-forward to define, deduction (and the *deduction theorem*) is commonly given in terms of *sets* of propositions. This formulation is suited to establishing strong completeness theorems and compactness theorems.

The presentation of deduction from a set follows the presentation of list deduction given for (:⊢).

1.10 Definition of Deduction

Just as deduction from a list $(:\vdash)$ can be defined in terms of $(:\rightarrow)$, deduction from a *set* of assumptions can be expressed in terms of $(:\vdash)$.

definition (**in** *Minimal-Logic*) *set-deduction* :: $'a$ *set* $\Rightarrow$ $'a$ $\Rightarrow$ *bool* (**infix** $\Vdash$ *60*)
where
$\Gamma \Vdash \varphi \equiv \exists\ \Psi.\ set(\Psi) \subseteq \Gamma \wedge \Psi :\vdash \varphi$

1.10.1 Interpretation as Minimal Logic

As in the case of $(:\vdash)$, the relation $(\Vdash)$ may be interpreted as a *proves* predicate for a fixed set of assumptions Γ.

The following lemma is given in order to establish this, which asserts that every minimal logic tautology $\vdash \varphi$ is also a tautology for $\Gamma \Vdash \varphi$.

lemma (**in** *Minimal-Logic*) *set-deduction-weaken*: $\vdash \varphi \Longrightarrow \Gamma \Vdash \varphi$
using *list-deduction-base-theory set-deduction-def* **by** *fastforce*

In the case of the empty set, the converse may be established.

lemma (**in** *Minimal-Logic*) *set-deduction-base-theory*: $\{\} \Vdash \varphi \equiv \vdash \varphi$
using *list-deduction-base-theory set-deduction-def* **by** *auto*

Next, a form of *modus ponens* is provided for $(\Vdash)$.

lemma (**in** *Minimal-Logic*) *set-deduction-modus-ponens*: $\Gamma \Vdash \varphi \rightarrow \psi \Longrightarrow \Gamma \Vdash \varphi \Longrightarrow \Gamma \Vdash \psi$
proof −
assume $\Gamma \Vdash \varphi \rightarrow \psi$
then obtain Φ **where** A: *set* $\Phi \subseteq \Gamma$ **and** B: $\Phi :\vdash \varphi \rightarrow \psi$
using *set-deduction-def* **by** *blast*
assume $\Gamma \Vdash \varphi$
then obtain Ψ **where** C: *set* $\Psi \subseteq \Gamma$ **and** D: $\Psi :\vdash \varphi$
using *set-deduction-def* **by** *blast*
from B D **have** Φ @ $\Psi :\vdash \psi$
using *list-deduction-cut-rule list-deduction-theorem* **by** *blast*
moreover from A C **have** *set* (Φ @ Ψ) $\subseteq \Gamma$ **by** *simp*
ultimately show *?thesis*
using *set-deduction-def* **by** *blast*
qed

context *Minimal-Logic* **begin**
interpretation *Set-Deduction-Logic*: *Minimal-Logic* $\lambda\ \varphi.\ \Gamma \Vdash \varphi$ $(\rightarrow)$
proof
fix φ ψ
show $\Gamma \Vdash \varphi \rightarrow \psi \rightarrow \varphi$ **by** (*metis Axiom-1 set-deduction-weaken*)
next
fix φ ψ χ
show $\Gamma \Vdash (\varphi \rightarrow \psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \chi$ **by** (*metis Axiom-2 set-deduction-weaken*)

next
  fix φ ψ
  show $\Gamma \Vdash \varphi \to \psi \Longrightarrow \Gamma \Vdash \varphi \Longrightarrow \Gamma \Vdash \psi$ **using** *set-deduction-modus-ponens* **by** *metis*
qed
end

1.11 The Deduction Theorem

The next result gives the deduction theorem for $(\Vdash)$.

theorem (**in** *Minimal-Logic*) *set-deduction-theorem*: *insert* φ $\Gamma \Vdash \psi = \Gamma \Vdash \varphi \to \psi$
proof –
 have $\Gamma \Vdash \varphi \to \psi \Longrightarrow$ *insert* φ $\Gamma \Vdash \psi$
  by (*metis set-deduction-def insert-mono list.simps*(*15*) *list-deduction-theorem*)
 moreover {
  assume *insert* φ $\Gamma \Vdash \psi$
  then obtain Φ **where** *set* $\Phi \subseteq$ *insert* φ Γ **and** $\Phi :\vdash \psi$
   using *set-deduction-def* **by** *auto*
  hence *set* (*removeAll* φ $\Phi) \subseteq \Gamma$ **by** *auto*
  moreover from ‹$\Phi :\vdash \psi$› **have** *removeAll* φ $\Phi :\vdash \varphi \to \psi$
   using *Modus-Ponens list-implication-removeAll list-deduction-def*
   by *blast*
  ultimately have $\Gamma \Vdash \varphi \to \psi$
   using *set-deduction-def* **by** *blast*
 }
 ultimately show *insert* φ $\Gamma \Vdash \psi = \Gamma \Vdash \varphi \to \psi$ **by** *metis*
qed

1.12 Monotonic Growth in Deductive Power

In contrast to the $(:\vdash)$ relation, the proof that the deductive power of $(\Vdash)$ grows monotonically with its assumptions may be fully automated.

theorem *set-deduction-monotonic*: $\Sigma \subseteq \Gamma \Longrightarrow \Sigma \Vdash \varphi \Longrightarrow \Gamma \Vdash \varphi$
 by (*meson dual-order.trans set-deduction-def*)

1.13 The Deduction Theorem Revisited

As a consequence of the fact that $[\![?\Sigma \subseteq ?\Gamma;\ ?\Sigma \Vdash ?\varphi]\!] \Longrightarrow ?\Gamma \Vdash ?\varphi$ automatically provable, the alternate *deduction theorem* where the discharged assumption is completely removed from the set of assumptions is just a consequence of the more conventional *insert* $?\varphi$ $?\Gamma \Vdash ?\psi = ?\Gamma \Vdash ?\varphi \to ?\psi$ and some basic set identities.

theorem (**in** *Minimal-Logic*) *alternate-set-deduction-theorem*:
 insert φ $\Gamma \Vdash \psi = \Gamma - \{\varphi\} \Vdash \varphi \to \psi$
 by (*metis insert-Diff-single set-deduction-theorem*)

1.14 Reflection

Just as in the case of ($:\vdash$), deduction from sets of assumptions makes true the *reflection principle* and is automatically provable.

theorem (**in** *Minimal-Logic*) *set-deduction-reflection*: $\varphi \in \Gamma \Longrightarrow \Gamma \Vdash \varphi$
by (*metis Set.set-insert*
list-implication.simps(1)
list-implication-Axiom-1
set-deduction-theorem
set-deduction-weaken)

1.15 The Cut Rule

The final principle of ($\Vdash$) presented is the *cut rule.*

First, the weak form of the rule is established.

theorem (**in** *Minimal-Logic*) *set-deduction-cut-rule*:
insert φ $\Gamma \Vdash \psi \Longrightarrow \Delta \Vdash \varphi \Longrightarrow \Gamma \cup \Delta \Vdash \psi$
proof −
assume *insert* φ $\Gamma \Vdash \psi$
hence $\Gamma \Vdash \varphi \to \psi$ **using** *set-deduction-theorem* **by** *auto*
hence $\Gamma \cup \Delta \Vdash \varphi \to \psi$ **using** *set-deduction-def* **by** *auto*
moreover assume $\Delta \Vdash \varphi$
hence $\Gamma \cup \Delta \Vdash \varphi$ **using** *set-deduction-def* **by** *auto*
ultimately show *?thesis* **using** *set-deduction-modus-ponens* **by** *metis*
qed

Another lemma is shown next in order to establish the strong form of the cut rule. The lemma shows the existence of a *covering list* of assumptions Ψ in the event some set of assumptions Δ proves everything in a finite set of assumptions Φ.

lemma (**in** *Minimal-Logic*) *finite-set-deduction-list-deduction*:
finite $\Phi \Longrightarrow$
$\forall\ \varphi \in \Phi.\ \Delta \Vdash \varphi \Longrightarrow$
$\exists \Psi.$ *set* $\Psi \subseteq \Delta \wedge (\forall \varphi \in \Phi.\ \Psi :\vdash \varphi)$
proof(*induct* Φ *rule*: *finite-induct*)
case *empty* **thus** *?case* **by** (*metis all-not-in-conv empty-subsetI set-empty*)
next
case (*insert* χ Φ)
assume $\forall \varphi \in \Phi.\ \Delta \Vdash \varphi \Longrightarrow \exists \Psi.$ *set* $\Psi \subseteq \Delta \wedge (\forall \varphi \in \Phi.\ \Psi :\vdash \varphi)$
and $\forall \varphi \in$ *insert* χ $\Phi.\ \Delta \Vdash \varphi$
hence $\exists \Psi.$ *set* $\Psi \subseteq \Delta \wedge (\forall \varphi \in \Phi.\ \Psi :\vdash \varphi)$ **and** $\Delta \Vdash \chi$ **by** *simp+*
then obtain Ψ_1 Ψ_2 **where**
set $(\Psi_1 \mathbin{@} \Psi_2) \subseteq \Delta$ **and**
$\forall \varphi \in \Phi.\ \Psi_1 :\vdash \varphi$ **and**
$\Psi_2 :\vdash \chi$
using *set-deduction-def* **by** *auto*
moreover from *this* **have** $\forall \varphi \in$ (*insert* χ Φ). $\Psi_1 \mathbin{@} \Psi_2 :\vdash \varphi$

by (*metis*
insert-iff
le-sup-iff
list-deduction-monotonic
order-refl set-append)
ultimately show *?case* **by** *blast*
qed

With $[\![finite\ ?\Phi;\ \forall\varphi\in ?\Phi.\ ?\Delta \Vdash \varphi]\!] \Longrightarrow \exists\Psi.\ set\ \Psi \subseteq\ ?\Delta \wedge (\forall\varphi\in ?\Phi.\ \Psi :\vdash \varphi)$ the strengthened form of the cut rule can be given.

theorem (**in** *Minimal-Logic*) *strong-set-deduction-cut-rule*:
$\Phi \cup \Gamma \Vdash \psi \Longrightarrow \forall\ \varphi \in \Phi.\ \Delta \Vdash \varphi \Longrightarrow \Gamma \cup \Delta \Vdash \psi$
proof $-$
assume $\Phi \cup \Gamma \Vdash \psi$
then obtain Σ **where**
A: $set\ \Sigma\ \subseteq \Phi \cup \Gamma$ **and**
B: $\Sigma :\vdash \psi$
using *set-deduction-def*
by *auto*+
obtain $\Phi'\ \Gamma'$ **where**
C: $set\ \Phi' = set\ \Sigma \cap \Phi$ **and**
D: $set\ \Gamma' = set\ \Sigma \cap \Gamma$
by (*metis inf-sup-aci*(*1*) *inter-set-filter*)+
then have $set\ (\Phi'\ @\ \Gamma') = set\ \Sigma$ **using** *A* **by** *auto*
hence *E*: $\Phi'\ @\ \Gamma' :\vdash \psi$ **using** *B list-deduction-monotonic* **by** *blast*
assume $\forall\ \varphi \in \Phi.\ \Delta \Vdash \varphi$
hence $\forall\ \varphi \in set\ \Phi'.\ \Delta \Vdash \varphi$ **using** *C* **by** *auto*
from *this* **obtain** Δ' **where** $set\ \Delta' \subseteq \Delta$ **and** $\forall\ \varphi \in set\ \Phi'.\ \Delta' :\vdash \varphi$
using *finite-set-deduction-list-deduction* **by** *blast*
with *strong-list-deduction-cut-rule D E*
have $set\ (\Gamma'\ @\ \Delta') \subseteq \Gamma \cup \Delta$ **and** $\Gamma'\ @\ \Delta' :\vdash \psi$ **by** *auto*
thus *?thesis* **using** *set-deduction-def* **by** *blast*
qed

1.16 Maximally Consistent Sets For Minimal Logic

definition (**in** *Minimal-Logic*)
Formula-Consistent :: $'a \Rightarrow\ 'a\ set \Rightarrow bool$ (*-–Consistent* - [*100*] *100*)
where [*simp*]: $\varphi{-}Consistent\ \Gamma \equiv\ \sim (\Gamma \Vdash \varphi)$

lemma (**in** *Minimal-Logic*) *Formula-Consistent-Extension*:
assumes $\varphi{-}Consistent\ \Gamma$
shows $(\varphi{-}Consistent\ insert\ \psi\ \Gamma) \vee (\varphi{-}Consistent\ insert\ (\psi \to \varphi)\ \Gamma)$
proof $-$
{
assume $\sim \varphi{-}Consistent\ insert\ \psi\ \Gamma$
hence $\Gamma \Vdash \psi \to \varphi$
using *set-deduction-theorem*
unfolding *Formula-Consistent-def*

```
      by simp
    hence φ−Consistent insert (ψ → φ) Γ
     by (metis Un-absorb assms Formula-Consistent-def set-deduction-cut-rule)
  }
  thus ?thesis by blast
qed

definition (in Minimal-Logic)
  Formula-Maximally-Consistent-Set
    :: 'a ⇒ 'a set ⇒ bool (-−MCS - [100] 100)
    where
      [simp]: φ−MCS Γ ≡ (φ−Consistent Γ) ∧ (∀ ψ. ψ ∈ Γ ∨ (ψ → φ) ∈ Γ)

theorem (in Minimal-Logic) Formula-Maximally-Consistent-Extension:
  assumes φ−Consistent Γ
  shows ∃ Ω. (φ−MCS Ω) ∧ Γ ⊆ Ω
proof −
  let ?Γ-Extensions = {Σ. (φ−Consistent Σ) ∧ Γ ⊆ Σ}
  have ∃ Ω ∈ ?Γ-Extensions. ∀Σ ∈ ?Γ-Extensions. Ω ⊆ Σ ⟶ Σ = Ω
  proof (rule subset-Zorn)
    fix 𝒞 :: 'a set set
    assume subset-chain-𝒞: subset.chain ?Γ-Extensions 𝒞
    hence 𝒞: ∀ Σ ∈ 𝒞. Γ ⊆ Σ ∀ Σ ∈ 𝒞. φ−Consistent Σ
      unfolding subset.chain-def by blast+
    show ∃ Ω ∈ ?Γ-Extensions. ∀ Σ ∈ 𝒞. Σ ⊆ Ω
    proof cases
      assume 𝒞 = {} thus ?thesis using assms by blast
    next
      let ?Ω = ⋃ 𝒞
      assume 𝒞 ≠ {}
      hence Γ ⊆ ?Ω by (simp add: 𝒞(1) less-eq-Sup)
      moreover have φ−Consistent ?Ω
      proof −
        {
          assume ~ φ−Consistent ?Ω
          then obtain ω where ω: finite ω ω ⊆ ?Ω ~ φ−Consistent ω
            unfolding Formula-Consistent-def
                      set-deduction-def
            by auto
          from ω(1) ω(2) have ∃ Σ ∈ 𝒞. ω ⊆ Σ
          proof (induct ω rule: finite-induct)
            case empty thus ?case using ‹𝒞 ≠ {}› by blast
          next
            case (insert ψ ω)
            from this obtain Σ₁ Σ₂ where
              Σ₁: ω ⊆ Σ₁ Σ₁ ∈ 𝒞 and
              Σ₂: ψ ∈ Σ₂ Σ₂ ∈ 𝒞
              by auto
            hence Σ₁ ⊆ Σ₂ ∨ Σ₂ ⊆ Σ₁
```

```
          using subset-chain-𝒞
          unfolding subset.chain-def
          by blast
        hence (insert ψ ω) ⊆ Σ₁ ∨ (insert ψ ω) ⊆ Σ₂
          using Σ₁ Σ₂ by blast
        thus ?case using Σ₁ Σ₂ by blast
      qed
      hence ∃ Σ ∈ 𝒞. (φ−Consistent Σ) ∧ ~ (φ−Consistent Σ)
        using 𝒞(2) ω(3)
        unfolding Formula-Consistent-def
                  set-deduction-def
        by auto
      hence False by auto
    }
    thus ?thesis by blast
  qed
  ultimately show ?thesis by blast
 qed
qed
then obtain Ω where Ω: Ω ∈ ?Γ-Extensions
                     ∀Σ ∈ ?Γ-Extensions. Ω ⊆ Σ ⟶ Σ = Ω by auto+
{
  fix ψ
  have (φ−Consistent insert ψ Ω) ∨ (φ−Consistent insert (ψ → φ) Ω)
       Γ ⊆ insert ψ Ω
       Γ ⊆ insert (ψ → φ) Ω
    using Ω(1) Formula-Consistent-Extension Formula-Consistent-def
    by auto
  hence insert ψ Ω ∈ ?Γ-Extensions
        ∨ insert (ψ → φ) Ω ∈ ?Γ-Extensions
    by blast
  hence ψ ∈ Ω ∨ (ψ → φ) ∈ Ω using Ω(2) by blast
}
thus ?thesis
  using Ω(1)
  unfolding Formula-Maximally-Consistent-Set-def
  by blast
qed

lemma (in Minimal-Logic) Formula-Maximally-Consistent-Set-reflection:
  φ−MCS Γ ⟹ ψ ∈ Γ = Γ ⊩ ψ
proof −
  assume φ−MCS Γ
  {
    assume Γ ⊩ ψ
    moreover from ‹φ−MCS Γ› have ψ ∈ Γ ∨ (ψ → φ) ∈ Γ ~ Γ ⊩ φ
      unfolding Formula-Maximally-Consistent-Set-def Formula-Consistent-def
      by auto
    ultimately have ψ ∈ Γ
```

using *set-deduction-reflection set-deduction-modus-ponens*
by *metis*
}
thus $\psi \in \Gamma = \Gamma \Vdash \psi$
using *set-deduction-reflection*
by *metis*
qed

theorem (**in** *Minimal-Logic*)
Formula-Maximally-Consistent-Set-implication-elimination:
assumes $\varphi{-}MCS\ \Omega$
shows $(\psi \to \chi) \in \Omega \Longrightarrow \psi \in \Omega \Longrightarrow \chi \in \Omega$
using *assms*
Formula-Maximally-Consistent-Set-reflection
set-deduction-modus-ponens
by *blast*

end

2 Combinatory Logic

theory *Combinators*
imports *./Minimal-Logic*
begin

2.1 Definitions

Combinatory logic, following Curry (TODO: citeme), can be formulated as follows.

datatype *Var* = *Var nat* ($\mathcal{X}$)

datatype *SKComb* =
Var-Comb Var ($\langle$-$\rangle$ [*100*] *100*)
| *S-Comb* (*S*)
| *K-Comb* (*K*)
| *Comb-App SKComb SKComb* (**infixl** $\cdot$ *75*)

Note that in addition to *S* and *K* combinators, *SKComb* provides terms for *variables*. This is helpful when studying λ-abstraction embedding.

2.2 Typing

The fragment of the *SKComb* types without *Var-Comb* terms can be given *simple types*:

datatype *'a Simple-Type* =
Atom 'a ($\{\!|$ - $|\!\}$ [*100*] *100*)
| *To 'a Simple-Type 'a Simple-Type* (**infixr** $\Rightarrow$ *70*)

inductive
Simply-Typed-SKComb
  $:: SKComb \Rightarrow {}'a\ Simple\text{-}Type \Rightarrow bool$ (**infix** :: *65*)
where
  $S\text{-}type : S :: (\varphi \Rightarrow \psi \Rightarrow \chi) \Rightarrow (\varphi \Rightarrow \psi) \Rightarrow \varphi \Rightarrow \chi$
| $K\text{-}type : K :: \varphi \Rightarrow \psi \Rightarrow \varphi$
| $Application\text{-}type : E_1 :: \varphi \Rightarrow \psi \Longrightarrow E_2 :: \varphi \Longrightarrow E_1 \cdot E_2 :: \psi$

2.3 Lambda Abstraction

Here a simple embedding of the λ-calculus into combinator logic is presented.

The SKI embedding below is originally due to David Turner [1].

Abstraction over combinators where the abstracted variable is not free are simplified using the K combinator.

primrec *free-variables-in-SKComb* $:: SKComb \Rightarrow Var\ set\ (free_{SK})$
 where
  $free_{SK}\ (\langle x \rangle) = \{x\}$
| $free_{SK}\ S = \{\}$
| $free_{SK}\ K = \{\}$
| $free_{SK}\ (E_1 \cdot E_2) = (free_{SK}\ E_1) \cup (free_{SK}\ E_2)$

primrec *Turner-Abstraction*
 $:: Var \Rightarrow SKComb \Rightarrow SKComb$ ($\boldsymbol{\lambda}$-. - [*90*,*90*] *90*)
 where
  $abst\text{-}S$: $\boldsymbol{\lambda}x.\ S = K \cdot S$
| $abst\text{-}K$: $\boldsymbol{\lambda}x.\ K = K \cdot K$
| $abst\text{-}var$: $\boldsymbol{\lambda}x.\ \langle y \rangle = (if\ x = y\ then\ S \cdot K \cdot K\ else\ K \cdot \langle y \rangle)$
| $abst\text{-}app$:
   $\boldsymbol{\lambda}\ x.\ (E_1 \cdot E_2) = (if\ (x \in free_{SK}\ (E_1 \cdot E_2))$
      $then\ S \cdot (\boldsymbol{\lambda}\ x.\ E_1) \cdot (\boldsymbol{\lambda}\ x.\ E_2)$
      $else\ K \cdot (E_1 \cdot E_2))$

2.4 Common Combinators

This section presents various common combinators. Some combinators are simple enough to express in using S and K, however others are more easily expressed using λ-abstraction. TODO: Cite Haskell Curry's PhD thesis.

A useful lemma is the type of the *identity* combinator, designated by I in the literature.

lemma *Identity-type*: $S \cdot K \cdot K :: \varphi \Rightarrow \varphi$
 using *K-type S-type Application-type* **by** *blast*

Another significant combinator is the combinator, which corresponds to `flip` in Haskell.

lemma *C-type*:
$\boldsymbol{\lambda}\ \mathcal{X}\ 1.\ \boldsymbol{\lambda}\ \mathcal{X}\ 2.\ \boldsymbol{\lambda}\ \mathcal{X}\ 3.\ (\langle \mathcal{X}\ 1\rangle \cdot \langle \mathcal{X}\ 3\rangle \cdot \langle \mathcal{X}\ 2\rangle)$
$:: (\varphi \Rightarrow \psi \Rightarrow \chi) \Rightarrow \psi \Rightarrow \varphi \Rightarrow \chi$
by (*simp*, *meson Identity-type Simply-Typed-SKComb.simps*)

Haskell also has a function `(.)`, which is referred to as the B combinator.

lemma *B-type*: $S \cdot (K \cdot S) \cdot K :: (\psi \Rightarrow \chi) \Rightarrow (\varphi \Rightarrow \psi) \Rightarrow \varphi \Rightarrow \chi$
by (*meson Simply-Typed-SKComb.simps*)

The final combinator given is the B combinator.

lemma *W-type*:
$\boldsymbol{\lambda}\ \mathcal{X}\ 1.\ \boldsymbol{\lambda}\ \mathcal{X}\ 2.\ (\langle \mathcal{X}\ 1\rangle \cdot \langle \mathcal{X}\ 2\rangle \cdot \langle \mathcal{X}\ 2\rangle) :: (\varphi \Rightarrow \varphi \Rightarrow \chi) \Rightarrow \varphi \Rightarrow \chi$
by (*simp*, *meson Identity-type Simply-Typed-SKComb.simps*)

2.5 The Curry Howard Correspondence

The (polymorphic) typing for a combinator X is given by the relation $X :: \varphi$.

Combinator types form an instance of minimal logic.

interpretation *Combinator-Minimal-Logic*: *Minimal-Logic* $\lambda\ \varphi.\ \exists\ X.\ X :: \varphi\ (\Rightarrow)$
proof qed (*meson Simply-Typed-SKComb.intros*)+

The minimal logic generated by combinator logic is *free* in the following sense: If $X :: \varphi$ holds for some combinator X then φ may be interpreted as logical consequence in any given minimal logic instance.

The fact that any valid type in combinator logic may be interpreted in minimal logic is a form of the *Curry-Howard correspondence*. TODO: Cite

primrec (**in** *Minimal-Logic*) *Simple-Type-interpretation*
$:: 'a\ \textit{Simple-Type} \Rightarrow 'a\ ((\!|\ \text{-}\ |\!)\ [50])$ **where**
$(\!|\ \textit{Atom}\ p\ |\!) = p$
$|\ (\!|\ \varphi \Rightarrow \psi\ |\!) = (\!|\ \varphi\ |\!) \to (\!|\ \psi\ |\!)$

lemma (**in** *Minimal-Logic*) *Curry-Howard-correspondence*:
$X :: \varphi \Longrightarrow \vdash (\!|\ \varphi\ |\!)$
by (*induct rule*: *Simply-Typed-SKComb.induct*,
(*simp add*: *Axiom-1 Axiom-2 Modus-Ponens*)+)

end

3 Kripke Semantics For Intuitionistic Logic

theory *Kripke-Semantics*
imports *Main*
./Combinators
begin

record ('a, 'b) *Kripke-Model* =
  R :: '$a \Rightarrow$ '$a \Rightarrow$ *bool*
  V :: '$a \Rightarrow$ '$b \Rightarrow$ *bool*

primrec
Intuitionistic-Kripke-Semantics
  :: ('a, 'b) *Kripke-Model* $\Rightarrow$ '$a \Rightarrow$ 'b *Simple-Type* $\Rightarrow$ *bool*
    (- - $\models$ - [*60*,*60*,*60*] *60*)
  where
    $(\mathfrak{M}\ x \models \{\!\!\{\ v\ \}\!\!\}) = (\exists\ w.\ (R\ \mathfrak{M})^{**}\ w\ x \wedge (V\ \mathfrak{M})\ w\ v)$
  $|\ (\mathfrak{M}\ x \models \varphi \Rightarrow \psi) = (\forall\ y.\ (R\ \mathfrak{M})^{**}\ x\ y \longrightarrow \mathfrak{M}\ y \models \varphi \longrightarrow \mathfrak{M}\ y \models \psi)$

lemma *Kripke-model-monotone*:
  $(R\ \mathfrak{M})^{**}\ x\ y \Longrightarrow \mathfrak{M}\ x \models \varphi \Longrightarrow \mathfrak{M}\ y \models \varphi$
  by (*induct* φ *arbitrary*: y; *simp*)
    (*meson rtranclp-trans*)+

lemma *Kripke-models-impl-flatten*:
  $\mathfrak{M}\ x \models \varphi \Rightarrow \psi \Rightarrow \chi =$
    $(\forall\ y.\ (R\ \mathfrak{M})^{**}\ x\ y \longrightarrow \mathfrak{M}\ y \models \varphi \longrightarrow \mathfrak{M}\ y \models \psi \longrightarrow \mathfrak{M}\ y \models \chi)$
  by (*rule iffI* ; *simp*)
    (*meson Kripke-model-monotone rtranclp-trans*)

lemma *Kripke-models-K*:
  $\mathfrak{M}\ x \models \varphi \Rightarrow \psi \Rightarrow \varphi$
  by (*meson Kripke-models-impl-flatten*)

lemma *Kripke-models-S*:
  $\mathfrak{M}\ x \models (\varphi \Rightarrow \psi \Rightarrow \chi) \Rightarrow (\varphi \Rightarrow \psi) \Rightarrow \varphi \Rightarrow \chi$
  by (*simp*, *meson rtranclp.rtrancl-refl rtranclp-trans*)

lemma *Kripke-models-Modus-Ponens*:
  $\mathfrak{M}\ x \models \varphi \Rightarrow \psi \Longrightarrow \mathfrak{M}\ x \models \varphi \Longrightarrow \mathfrak{M}\ x \models \psi$
  by *auto*

theorem *Combinator-Typing-Kripke-Soundness*:
  $X :: \varphi \Longrightarrow \mathfrak{M}\ x \models \varphi$
  by (*induct rule*: *Simply-Typed-SKComb.induct*)
    (*meson Kripke-models-S*, *meson Kripke-models-K*, *auto*)

lemma *Combinator-Typing-Kripke-Soundness-alt*:
  $\exists\ X\ .\ X :: \varphi \Longrightarrow \forall\ \mathfrak{M}\ x.\ \mathfrak{M}\ x \models \varphi$
  by (*meson Combinator-Typing-Kripke-Soundness*)

lemma *Kripke-Cont-Monad*:
  assumes $a \neq b$

and $p \neq q$
and $\mathfrak{M} = (\!|\ R = (\lambda\ x\ y.\ x = a \wedge y = b),\ V = (\lambda\ x\ y.\ x = b \wedge y = p)\ |\!)$
shows $\neg\ \mathfrak{M}\ a \models ((\{\!\!|\ p\ |\!\!\} \Rightarrow \{\!\!|\ q\ |\!\!\}) \Rightarrow \{\!\!|\ q\ |\!\!\}) \Rightarrow \{\!\!|\ p\ |\!\!\}$
proof $-$
have $\neg\ \mathfrak{M}\ b \models \{\!\!|\ p\ |\!\!\} \Rightarrow \{\!\!|\ q\ |\!\!\}$
$\neg\ \mathfrak{M}\ a \models \{\!\!|\ p\ |\!\!\} \Rightarrow \{\!\!|\ q\ |\!\!\}$
unfolding *assms*(*3*)
using *assms*(*1*) *assms*(*2*) **by** *auto*
hence $\forall\ x.\ (R\ \mathfrak{M})^{**}\ a\ x \longrightarrow \neg\ \mathfrak{M}\ x \models \{\!\!|\ p\ |\!\!\} \Rightarrow \{\!\!|\ q\ |\!\!\}$
unfolding *assms*(*3*)
by (*simp*, *metis* (*mono-tags*, *lifting*) *rtranclp.simps*)
hence $\mathfrak{M}\ a \models (\{\!\!|\ p\ |\!\!\} \Rightarrow \{\!\!|\ q\ |\!\!\}) \Rightarrow \{\!\!|\ q\ |\!\!\}$
by *fastforce*
moreover have $\neg\ \mathfrak{M}\ a \models \{\!\!|\ p\ |\!\!\}$
unfolding *assms*(*3*)
using *assms*(*1*) *converse-rtranclpE* **by** *fastforce*
ultimately show *?thesis*
by (*meson Kripke-models-Modus-Ponens*)
qed

lemma *no-extract*:
assumes $p \neq q$
shows $\nexists\ X\ .\ X :: ((\{\!\!|\ p\ |\!\!\} \Rightarrow \{\!\!|\ q\ |\!\!\}) \Rightarrow \{\!\!|\ q\ |\!\!\}) \Rightarrow \{\!\!|\ p\ |\!\!\}$
using *assms*
by (*metis*
Combinator-Typing-Kripke-Soundness
Kripke-Cont-Monad)

end

References

[1] D. A. Turner. Another algorithm for bracket abstraction. *Journal of Symbolic Logic*, 44(2):267–270, June 1979.