CS292C Final Project proof of type soundness

Daniel Zhang

March 16, 2020

1 Introduction

For this project I've decided to prove type soundness for and implement the Calculus of Constructions. Extensions of the Calculus of Constructions are used in Coq and Agda [2], so understanding it will help with understanding those systems. I will not be describing extensions such as inductive types or universes.

The Calculus of Constructions is a Pure Type System. Pure Type Systems have a set of sorts. The Calculus of Constructions has three sorts: terms, types and kinds (the type of types).

$\star$ is a constant of sort kind. Note that it is not the only kind. $\star \to \star$ is a different kind. This corresponds to the type of a type constructor: a function that takes in a type and outputs a type.

The notation $\lambda x : A.B$ is used both for a function that takes in a term and a function that takes in a type. $\Pi x : A.B$ is a dependent product. If x does not occur in B, it is the same as $A \to B$. When A is $\star$, it is equivalent to a universal type. [3]

An example of a term would be $\lambda T : *.(\lambda x : T.x)$. Types include expressions such as $\lambda x : \star.x$ and $\Pi x : \star.x$. An example of a kind would be $\Pi x : \star.\star$.

The Calculus of Constructions is the most powerful system in the Lambda Cube. Both types and terms can depend on types and terms. The Calculus of Constructions is known to be strongly-normalizing, but the proof is considered difficult [2].

2 Syntax

Expressions are considered equivalent up to α-conversion. Variables names are assumed to all be distinct (implementation uses de Bruijn indices, so substitution doesn't cause issues).

$$k \in \text{Const} \to \star \mid \square$$

$$e \in \text{Exp} \to k \in \text{Const} \mid x \in \text{Variable} \mid AB \mid \lambda x : A.B \mid \Pi x : A.B$$

This is the syntax for both terms and types.

$$v \in \text{Values} \to k \in \text{Const} \mid x \in \text{Variable} \mid \lambda x : A.B \mid \Pi x : A.B$$

where $A, B \in \text{Values}$.

3 Semantics

These rules are from [2], modifed to be deterministic.

$$\frac{A \to A'}{(\lambda x : A.B) \to (\lambda x : A'.B)} \ (LAM1)$$

$$\frac{B \to B'}{(\lambda x : A.B) \to (\lambda x : A.B')} \text{ if } A \in \text{Values} (LAM2)$$

$$\frac{A \to A'}{(\Pi x : A.B) \to (\Pi x : A'.B)} \ (PI1)$$

$$\frac{B \to B'}{(\Pi x : A.B) \to (\Pi x : A.B')} \text{ if } A \in \text{Values} (PI2)$$

$$\frac{A \to A'}{AB \to A'B} \text{ if APP3 cannot apply} (APP1)$$

$$\frac{B \to B'}{AB \to AB'} \text{ if } A \in \text{Values and APP3 cannot apply} (APP2)$$

$$\frac{}{(\lambda x : A.B)C \to B[x \mapsto C]} \ (APP3)$$

4 Substitution

$$k[x \mapsto D] = k$$

$$y[x \mapsto D] = \begin{cases} D & \text{if } x = y \\ y & \text{if } x \neq y \end{cases}$$

$$(AB)[x \mapsto D] = A[x \mapsto D]B[x \mapsto D]$$

$$(\lambda y : A.B)[x \mapsto D] = \lambda y : A[x \mapsto D].B[x \mapsto D]$$

$$(\Pi y : A.B)[x \mapsto D] = \Pi y : A[x \mapsto D].B[x \mapsto D]$$

$$(\Gamma, y : B)[x \mapsto D] = \Gamma[x \mapsto D], y : B[x \mapsto D]$$

5 Typing rules

$$\frac{}{\vdash \star : \Box}\ (STAR)$$

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}\ (VAR)$$

$$\frac{\Gamma \vdash B : C \quad \Gamma \vdash A : s}{\Gamma, x : A \vdash B : C}\ (WEAK)$$

$$\frac{\Gamma \vdash f : (\Pi x : A.B) \quad \Gamma \vdash a : A}{\Gamma \vdash fa : B[x \mapsto a]}\ (APP)$$

$$\frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash (\Pi x : A.B) : t}{\Gamma \vdash (\lambda x : A.b) : (\Pi x : A.B)}\ (LAM)$$

$$\frac{\Gamma \vdash A : s \quad \Gamma, x : A \vdash B : t \quad s, t \in \{\star, \Box\}}{\Gamma \vdash (\Pi x : A.B) : t}\ (PI)$$

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash B : s \quad A =_\beta B}{\Gamma \vdash a : B}\ (CONV)$$

$$\frac{}{(\lambda x : A.B)C =_\beta B[x \mapsto C]}\ (=_\beta LAM)$$

and its symmetric transitive closure.

These rules are taken from [3] with some modifications. The notation for substitution was changed to match what was used in this class. The version in the paper had a relation on s and t in rule (PI) to restrict to other type systems in the Lambda Cube. The eight systems of the Lambda Cube correspond to all choices of pairs of valid (s, t) in rule PI, where $(\star, \star)$ must be taken. The Calculus of Constructions allows all pairs of sorts on $\{\star, \Box\}$.

6 Proof of progress

Lemma 1. *Let e be a value. If $\Gamma \vdash e : \Pi x : A.B$, then $e = (\lambda x : C.D)$.*

Proof. Since e is a value, it must either be of the form $\lambda x : C.D$ or $\Pi x : C.D$. Let T the last type before trailing applications of CONV or WEAK: $\Gamma' \vdash e : T$ and $T =_\beta \Pi x.A : B$. If $e = \Pi x : C.D$, then $T \in \{\star, \Box\}$ since PI is the only rule that could be applied based on the form of e. But since neither of $\star$ or $\Box$ is β-equivalent to $\Pi x : C.D$, this is a contradiction. Therefore, $e = \lambda x : C.D$. □

Theorem 1. $(\forall e \in Exp)(\vdash e : \tau) \Rightarrow (e \in Values \vee (\exists e' \in Exp)(e \to e'))$

Proof. Use structural induction on e:

Case $e \in Const$: $e \in Values$

Case $e \in Variable$: A variable can only have a type if it is in the context. Since we are in the empty context, a variable cannot be well-typed. Thus, this case cannot happen.

Case $e = AB$:

- If A is not a value, then $A \to A'$ by the inductive hypothesis. $e \to e'$ where $e' = A'B$ (rule APP1).
- If A is a value but B is not a value, then $B \to B'$ by the inductive hypothesis. Thus, $e \to e'$ where $e' = AB'$ (rule APP2).
- If A and B are both values: After removing trailing application of CONV from the typing derivation for τ, we get $AB = \tau'$ where $\tau =_\beta \tau'$. The last rule used in the typing derivation for τ' is not CONV, so it must be APP. Thus, $A : \Pi x : C.D$. Then by Lemma 1, $A = (\lambda x : \tau.e_3)$, so $e = (\lambda x : \tau.e_3)B$. Finally, $e \to e'$ where $e' = e_3[x \mapsto B]$ (rule APP3).

Case $e = (\lambda x : A.B)$: If A is not a value, then $A \to A'$ by the inductive hypothesis. $e \to e'$ where $e' = (\lambda x : A'.B)$. Otherwise, if B is not a value, then $B \to B'$ by the inductive hypothesis. $e \to e'$ where $e' = (\lambda x : A.B')$. If both A and B are values, then $e \in Values$.

Case $e = (\Pi x : A.B)$: If A is not a value, then $A \to A'$ by the inductive hypothesis. $e \to e'$ where $e' = (\Pi x : A'.B)$. Otherwise, if B is not a value, then $B \to B'$ by the inductive hypothesis. $e \to e'$ where $e' = (\Pi x : A.B')$. If both A and B are values, then $e \in Values$.

This exhausts all the cases, so the theorem is true by induction.

□

7 Proof of preservation

Compared to simply-typed lambda calculus, a major differences in the proving the Substitution Lemma is the type of a variable can depend on the value of another variable. This means order of variables in the context matters. Also, we must use the version of substitution that allows substituting arbitrary terms because rule APP requires it.

Lemma 2 (Substitution Lemma for Terms)**.** *If y is fresh in D,*

$$e[x \mapsto D][y \mapsto C[x \mapsto D]] = e[y \mapsto C][x \mapsto D]$$

Proof. The proof is by induction on the structure of e.

Case $e = k$: $k[x \mapsto D][y \mapsto C[x \mapsto D]] = k = k[y \mapsto C][x \mapsto D]$, as desired.

Case $e = AB$:

$$\begin{aligned}(AB)[x \mapsto D][y \mapsto C[x \mapsto D]] &= A[x \mapsto D][y \mapsto C[x \mapsto D]]B[x \mapsto D][y \mapsto C[x \mapsto D]] \\ &= A[y \mapsto C][x \mapsto D]B[y \mapsto C][x \mapsto D] \\ &= (AB)[y \mapsto C][x \mapsto D]\end{aligned}$$

Using the inductive hypothesis on A and B.

Case $e = (\lambda x : A.B)$: identical to AB.

Case $e = (\Pi x : A.B)$: identical to AB.

Case $e = x$:

$$\begin{aligned} x[x \mapsto D][y \mapsto C[x \mapsto D]] &= D[y \mapsto C[x \mapsto D]] \\ &= D \\ &= x[x \mapsto D] \\ &= x[y \mapsto C][x \mapsto D] \end{aligned}$$

Case $e = y$ where $y \neq x$:

$$\begin{aligned} y[x \mapsto D][y \mapsto C[x \mapsto D]] &= y[y \mapsto C[x \mapsto D]] \\ &= C[x \mapsto D] \\ &= y[y \mapsto C][x \mapsto D] \end{aligned}$$

□

Lemma 3 (Substitution preserves β-equivalence)**.**

$$\textit{If } A =_\beta B, \textit{ then } A[x \mapsto D] =_\beta B[x \mapsto D]$$

Proof. It suffices to show that $((\lambda y : A.B)C)[x \mapsto D] =_\beta (B[y \mapsto C])[x \mapsto D]$

$$\begin{aligned} ((\lambda y : A.B)C)[x \mapsto D] &= ((\lambda y : A.B)[x \mapsto D]C[x \mapsto D]) \\ &= ((\lambda y : A[x \mapsto D].B[x \mapsto D])C[x \mapsto D]) \\ &=_\beta B[x \mapsto D][y \mapsto C[x \mapsto D]] \\ &= B[y \mapsto C][x \mapsto D] \end{aligned}$$

since y is fresh in D (Substitution Lemma for Terms). □

Lemma 4 (Substitution Lemma)**.**

$$\textit{If } \Gamma, x : A, \Delta \vdash B : C \textit{ and } \Gamma \vdash D : A, \textit{ then } \Gamma, \Delta[x \mapsto D] \vdash B[x \mapsto D] : C[x \mapsto D]$$

This is Lemma 5.2.1 in [1].

Proof. Use induction on the length of the derivation of $\Gamma, x : A, \Delta, \vdash B : C$.

Abbreviate $M[x \mapsto D]$ as M^*. Note if x is fresh in M, then $M = M^*$.

Look at the last rule used in the derivation of $\Gamma, x : A, \Delta, \vdash B : C$.

Case STAR: This case cannot apply because STAR only applies in the empty context, but the context contains x.

Case WEAK: If $\Delta =<>$, the last rule was

$$\frac{\Gamma \vdash B : C \quad \Gamma \vdash A : s}{\Gamma, x : A \vdash B : C}$$

Since x is fresh in B and C, $B = B^*$, and $C = C^*$. Thus,

$$\Gamma \vdash B^* : C^*$$

as desired. On the other hand, if $\Delta = \Delta', y : E$, the last rule was

$$\frac{\Gamma, x : A, \Delta' \vdash B : C \quad \Gamma \vdash E : s}{\Gamma, x : A, \Delta', y : E \vdash B : C}$$

By the inductive hypothesis, $\Gamma, \Delta'^* \vdash B^* : C^*$. Also, since x is fresh in E, $E = E^*$. By rule WEAK,

$$\frac{\Gamma, \Delta'^* \vdash B^* : C^* \quad \Gamma \vdash E^* : s}{\Gamma, \Delta'^*, y : E^* \vdash B^* : C^*}$$

Since $\Delta^* = \Delta'^*, y : E^*$,

$$\Gamma, \Delta^* \vdash B^* : C^*$$

as desired.

Case VAR: If $\Delta =<>$, the last rule was

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}$$

Since $\Gamma \vdash D : A$, $x^* = D$, and $A = A^*$,

$$\Gamma \vdash x^* : A^*$$

as desired. On the other hand, if $\Delta = \Delta', y : E$, the last rule was

$$\frac{\Gamma, x : A, \Delta' \vdash E : s}{\Gamma, x : A, \Delta', y : E \vdash y : E}$$

By the inductive hypothesis,

$$\Gamma, \Delta'^* \vdash E^*, s^*$$

Now apply rule VAR:

$$\frac{\Gamma, \Delta'^* \vdash E^*, s^*}{\Gamma, \Delta'^*, y : E^* \vdash y : E^*}$$

Since $\Delta^* = \Delta'^*, y : E^*$ and $y = y^*$,

$$\Gamma, \Delta^* \vdash y^* : E^*$$

as desired.

Case PI: The last rule was

$$\frac{\Gamma, x : A, \Delta \vdash A : s \quad \Gamma, x : A, \Delta, y : E \vdash F : t}{\Gamma, x : A, \Delta \vdash (\Pi y : E.F) : t}$$

By the inductive hypothesis, $\Gamma, \Delta^* \vdash A^* : s^*$ and $\Gamma, \Delta^*, y : E^* \vdash F^* : t^*$ Applying rule PI, we get

$$\frac{\Gamma, \Delta^* \vdash A^* : s^* \quad \Gamma, \Delta^*, y : E^* \vdash F^* : t^*}{\Gamma, \Delta^* \vdash (\Pi y : E^* : F^*) : t^*}$$

as desired

Case LAM: The last rule was

$$\frac{\Gamma, x : A, \Delta, y : E \vdash f : F \quad \Gamma, x : A, \Delta \vdash (\Pi y : E.F) : t}{\Gamma, x : A, \Delta \vdash (\lambda y : E.f) : (\Pi y : E.F)}$$

By the inductive hypothesis, $\Gamma, \Delta^*, y : E^* \vdash f^* : F^*$ and $\Gamma, \Delta^* \vdash (\Pi y : E^*.F^*) : t^*$ Applying rule LAM,

$$\frac{\Gamma, \Delta^*, y : E^* \vdash f^* : F^* \quad \Gamma, \Delta^* \vdash (\Pi y : E^*.F^*) : t^*}{\Gamma, \Delta^* \vdash (\lambda y : E^*, f^*) : (\Pi y : E^* : F^*)}$$

as desired.

Case CONV: The last rule was

$$\frac{\Gamma, x : A, \Delta \vdash e : E \quad \Gamma, x : A, \Delta \vdash F : s \quad E =_\beta F}{\Gamma, x : A, \Delta \vdash e : F}$$

By the inductive hypothesis, $\Gamma, \Delta^* \vdash e^* : E^*$ and $\Gamma, \Delta^* \vdash F^* : s^*$. Also, $E^* =_\beta F^*$ since substitution preserves β-equivalence. By rule LAM,

$$\frac{\Gamma, \Delta^* \vdash e^* : E^* \quad \Gamma, \Delta^* \vdash F^* : s^* \quad E^* =_\beta F^*}{\Gamma, \Delta^* \vdash e^* : F^*}$$

This exhausts all the cases, so the lemma is true by induction. □

Theorem 2 (Preservation)**.** $(\Gamma \vdash A : B) \wedge (A \to A') \Rightarrow (\Gamma \vdash A' : B)$

This is based on Theorem 5.2.15 in [1].

Proof. Because the types of variables can contains other variables, we cannot just prove this by induction. Instead, we need to prove two statements simultaneously by induction on generation of $\Gamma \vdash A : B$:

$$(\Gamma \vdash e : \tau) \wedge (e \to e') \Rightarrow (\Gamma \vdash e' : \tau)$$

$$(\Gamma \vdash e : \tau) \wedge (\Gamma \to \Gamma') \Rightarrow (\Gamma' \vdash e : \tau)$$

where $\Gamma \to \Gamma'$ if $\Gamma = x_1 : A_1, \ldots x_n : A_n$ and $\Gamma' = x_1 : A_1' \ldots x_n : A_n'$ where $A_i \to A_i'$ for exactly one i and for all $j \neq i$ $A_j =_\beta A_j'$. In other words, the type of exactly one variable in the context takes a step.

Case STAR: This case cannot occur because neither the context nor the term may take a step.

Case VAR:

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}$$

Since x cannot take a step, we only need to handle $(\Gamma, x : A) \to (\Gamma, x : A)'$ This can happen in two ways:

If $\Gamma \to \Gamma'$: By the inductive hypothesis, $\Gamma' \vdash A : s$. Thus,

$$\frac{\Gamma' \vdash A : s}{\Gamma', x : A \vdash x : A}$$

as desired.

If $A \to A'$: By the inductive hypothesis, $\Gamma \vdash A' : s$. Thus,

$$\frac{\Gamma \vdash A' : s}{\Gamma, x : A' \vdash x : A'}$$

as desired.

Case WEAK:

$$\frac{\Gamma \vdash B : C \quad \Gamma \vdash A : s}{\Gamma, x : A \vdash B : C}$$

First, consider the case where the term takes a step: $B \to B'$. By the inductive hypothesis, $\Gamma \vdash B' : C$. By WEAK, $\Gamma, x : A \vdash B' : C$, as desired. Now consider if the context takes a step. There are two cases. If $\Gamma \to \Gamma'$, then by the inductive hypothesis, $\Gamma' \vdash B : C$ and $\Gamma' \vdash A : s$. By WEAK, $\Gamma' \vdash B : C$. If $A \to A'$, then by the inductive hypothesis, $\Gamma \vdash A' : s$. By WEAK, $\Gamma, x : A' \vdash B : C$, as desired.

Case APP:

$$\frac{\Gamma \vdash f : (\Pi : A.B) \quad \Gamma \vdash a : A}{\Gamma \vdash fa : B[x \mapsto a]}$$

If $\Gamma \to \Gamma'$, by the inductive hypothesis, $\Gamma' \vdash f : (\Pi : A.B)$ and $\Gamma' \vdash a : A$. Thus,

$$\Gamma \vdash fa : B[x \mapsto a]$$

as desired.

There are three ways for the term to take a step.

If $e' = f'a$ where $f \to f'$ (APP1), then by the inductive hypothesis, $\Gamma \vdash f' : (\Pi : A.B)$, so by APP, $\Gamma \vdash f'a : B[x \mapsto a]$, as desired. If $e' = fa'$ where $a \to a'$ (APP2), then by the inductive hypothesis, $\Gamma \vdash a' : A$, so by APP, $\Gamma \vdash fa' : B[x \mapsto a']$, as desired.

If $e' = b[x \mapsto a]$ where $f = (\lambda x : A.b)$ (APP3), then by rules APP and LAM (and possible WEAK and CONV), we have

$$\dfrac{\dfrac{\dfrac{\dfrac{\Delta, x : A \vdash b : B \quad \Delta \vdash (\Pi x : A.B) : t}{\Delta \vdash (\lambda x : A.b) : (\Pi x : A.B)}}{\cdots}}{\Gamma \vdash (\lambda x : A.b) : (\Pi x : A.B)} \qquad \Gamma \vdash x : A}{\Gamma \vdash (\lambda x : A.b)a : B[x \mapsto a]}$$

We can add back variables to Δ to get $\Gamma, x : A \vdash b : B$. Also, $\Gamma \vdash a : A$. By the Substitution Lemma. $\Gamma \vdash b[x \mapsto a] : B[x \mapsto a]$, as desired.

Case LAM: By rules LAM and PI (and possible WEAK and CONV).

$$\dfrac{\Gamma, x : A \vdash b : B \qquad \dfrac{\dfrac{\Delta \vdash A : s \quad \Delta, x : A \vdash B : u}{\Delta \vdash (\Pi x : A.B) : u}}{\cdots}}{\Gamma \vdash (\lambda x : A.b) : (\Pi x : A.B)}$$

There are two ways for the term to take a step. If $e' = (\lambda x : A'.b)$ where $A \to A'$ (LAM1), then by the inductive hypothesis, $\Gamma, x : A' \vdash b : B$, $\Delta \vdash A' : s$ and $\Delta, x : A' \vdash B : t$. Thus,

$$\dfrac{\Gamma, x : A' \vdash b : B \qquad \dfrac{\dfrac{\Delta \vdash A' : s \quad \Delta, x : A' \vdash B : u}{\Delta \vdash (\Pi x : A'.B) : u}}{\cdots}}{\Gamma \vdash (\lambda x : A'.b) : (\Pi x : A'.B)}$$

as desired.

If $e' = (\lambda x : A.b')$ where $b \to b'$ (LAM2), then by the inductive hypothesis, $\Gamma, x : A \vdash b' : B$, Thus,

$$\dfrac{\Gamma, x : A \vdash b' : B \quad \Gamma \vdash (\Pi x : A.B) : t}{\Gamma \vdash (\lambda x : A.b') : (\Pi x : A.B)}$$

as desired.

If $\Gamma \to \Gamma'$, by the inductive hypothesis, $\Gamma', x : A \vdash b : B$ and $\Gamma' \vdash (\Pi x : A.B) : t$.

Thus,

$$\dfrac{\Gamma', x : A \vdash b : B \quad \Gamma' \vdash (\Pi x : A.B) : t}{\Gamma' \vdash (\lambda x : A.b) : (\Pi x : A.B)}$$

as desired.

Case PI: $e = (\Pi x : A_1.A_2)$.

$$\dfrac{\Gamma \vdash A_1 : s \quad \Gamma, x : A_1 \vdash A_2 : t}{\Gamma \vdash (\Pi x : A_1.A_2) : t}$$

Suppose $e \to e'$. This can happen in two ways.

If $e' = A_1' A_2$ where $A_1 \to A_1'$ (rule PI1): By the inductive hypothesis, $\Gamma \vdash A_1' : s$, $\Gamma, x : A_1' \vdash A_2 : t$. Thus, by rule PI,

$$\dfrac{\Gamma \vdash A_1' : s \quad \Gamma, x : A_1' \vdash A_2 : t}{\Gamma \vdash (\Pi x : A_1'.A_2) : t}$$

as desired.

If $e' = A_1 A_2'$ where $A_2 \to A_2'$ (rule PI2): By the inductive hypothesis, $\Gamma, x : A_1 \vdash A_2' : t$. Thus, by rule PI,

$$\frac{\Gamma \vdash A_1 : s \quad \Gamma, x : A_1 \vdash A_2' : t}{\Gamma \vdash (\Pi x : A_1 . A_2') : t}$$

as desired.

Suppose $\Gamma \to \Gamma'$. By the inductive hypothesis, $\Gamma' \vdash A_1 : s$, $\Gamma', x : A_1 \vdash A_2 : t$. Thus, by rule PI,

$$\frac{\Gamma' \vdash A_1 : s \quad \Gamma', x : A_1 \vdash A_2 : t}{\Gamma' \vdash (\Pi x : A_1 . A_2) : t}$$

as desired.

Case CONV:

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash B : s \quad A =_\beta B}{\Gamma \vdash a : B}$$

Suppose $a \to a'$.

By the inductive hypothesis, $\Gamma \vdash a' : A$, so by rule CONV,

$$\frac{\Gamma \vdash a' : A \quad \Gamma \vdash B : s \quad A =_\beta B}{\Gamma \vdash a' : B}$$

as desired.

Suppose $\Gamma \to \Gamma'$. By the inductive hypothesis, $\Gamma' \vdash a : A$ and $\Gamma' \vdash B : s$, so by rule CONV,

$$\frac{\Gamma \vdash a' : A \quad \Gamma \vdash B : s \quad A =_\beta B}{\Gamma \vdash a' : B}$$

as desired. □

References

[1] H. P. Brendregt. Lambda calculi with types. In D. M. Gabbay S. Abramsky and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science vol. 2*, pages 117–309. Oxford University Press, Nijmegen, 1992.

[2] C. Casinghino. Strong normalization for the calculus of constructions. `https://prosecco.gforge.inria.fr/personal/hritcu/temp/snforcc.pdf`, 2010. Online; accessed: 2020-02-29.

[3] E. Meijer S. P. Jones. Henk: a typed intermediate language. `https://www.microsoft.com/en-us/research/wp-content/uploads/1997/01/henk.pdf`, 1997. Online; accessed: 2020-02-28.