Modelling the US Constitution to establish constitutional dictatorship

Valeria Zahoransky[1,2][0000−0002−6304−9720] and Christoph Benzmüller[2,3,⋆][0000−0002−3392−3093]

[1] University of Oxford, Oxford, Great Britain
valeria.zahoransky@maths.ox.ac.uk
[2] Freie Universität Berlin, Berlin, Germany
c.benzmueller@fu-berlin.com
[3] University of Luxembourg, Esch-sur-Alzette, Luxembourg

Abstract. We present a case example on how to conduct computer aided reasoning on legal texts. The basis is an anecdote of Kurt Gödel's citizenship hearing in which he claimed that the US Constitution allowed for the erection of a dictatorship. We shall model relevant parts of the US Constitution and conduct reasoning on them. This is done using the language of classical Higher Order Logic (HOL) and proof assistant Isabelle/HOL.

Keywords: legal reasoning · US Constitution · higher order logic.

1 Introduction

There is an infamous anecdote on how logician Kurt Gödel tried to explain a fault of the US Constitution to the judge hearing him for citizenship. When preparing for the hearing Gödel found that the US Constitution allowed for the introduction of a constitutional dictatorship. He set out to explain this to the judge once the discussion turned towards the governmental system of the United States. The judge was not interested in hearing Gödel's argument but did grant him the US citizenship. [3,10,12,14].

In the following we shall model an argument for installing lawful dictatorship on the basis of the US Constitution. It is not, however, Gödel's own argument, but rather one suggested by legal scholar Guerra-Pujol [8]. Gödel's original argument was not to be found in letters to his mother [6], letters to his colleagues [4] or in character witness Oskar Morgenstern's account of the hearing [10]. Morgenstern does mention conversations with Gödel on the alleged fault in the account and his diary but does not go into detail about Gödel's reasoning [10,11].

We will model Guerra-Pujol's argument with the language of Higher Order Logic (HOL) using Isabelle/HOL [13]. Throughout this paper we will present Isabelle code together with explanations of what the code does.

⋆ Benzmüller received support by VolkswagenStiftung under grant *Consistent Rational Argumentation in Politics* (CRAP).

In the following *the Constitution* and *US Constitution* shall be used interchangeably.

2 Developing a model

2.1 On the argument used

Below, we outline the argument as provided by [8].

The constitution does not allow for the direct installation of a dictatorship, since dictatorship requires the consolidation of legislative, executive and judicial powers in one person or institution [9]. This is not possible due to the separation of powers as set out in U.S. Const. Art.I-III. In order to allow for this kind of consolidation of powers the Constitution has to be amended in a two-step process. First, an amendment that changes Art.V has to be introduced and secondly an amendment that actually installs dictatorship by consolidating power in one person or institution.

Art.V needs to be amended since it regulates the amendment process and protects some articles from being amended altogether, such as U.S. Const. Art.I, §3., cl.1. and U.S. Const. Amend.XXVII which ensure that each state has two votes in Senate. Directly introducing an amendment that would abolish the distribution of powers and thus strip the states of their suffrage rights would not be constitutional. One can however remove the protection of certain articles from Art.V with a first amendment, *amd1*, and then introduce dictatorship with a second amendment, *amd2*. This is constitutional since Art.V does not protect itself.

Consequently, the outline for our model is as follows:

time instance	t_1	t_2	t_3
Constitution state	Current Constitution	Constitution of t_1 + *amd1*	Constitution of t_2 + *amd2*
	Distribution of powers	Distribution of powers	No distribution of powers
	No dictatorship	No dictatorship	Dictatorship
	Proposal of *amd1*	Proposal of *amd2*	

2.2 Modelling the argument

On representing time As seen above, we want to represent the changes of the Constitution over different instances of time.

We choose to do this via temporal logic. Generally, such a logic would be expressed by a set T of instances of time and a precedence relation $\prec$ on $T \times T$, such that $\prec$ is both irreflexive and transitive [7]. We shall not require a relation to be transitive, however. Neither will we use modal operators to express that certain events will *always* occur in the future or that an event will occur *at some point* in the future. The same goes for events in the past. We only require an operator X that refers to the immediate successor of an instance of time. The operator is denoted by X for the "x" in "next".

To understand why this is sensible in our case, consider above given table which outlines what we would like to express. Assume that $T = \{t_1, t_2, t_3\}$ and $t_1 \prec t_2$,$t_2 \prec t_3$ and $t_i \not\prec t_j$ for all other combinations of t_i and t_j in T :

The basis for changes in t_2 is set out with *amd1* at t_1. Likewise the basis for changes in t_3 is set out with *amd2* at t_2. At each $t_i \in T$ the furthest we look into the future is the immediate successor, thus we do not need $\prec$ to be transitive.

In addition to it not being necessary, there is another reason to omit transitivity as requirement for the precedence relation. For a formula φ, we would like $X\ \varphi$ to be valid at point t iff for any t', s.t. $t \prec t'$, holds: φ is valid at t'. If $\prec$ were transitive, then $X\ \varphi$ would not mean "φ is valid at the next instance after t", but "φ is valid at all instances after t". If not used very carefully, this could easily lead to inconsistencies. After all, amendments do not necessarily stay valid once ratified.[4] Since we do not need a transitive relation $\prec$, it is advisable to avoid it altogether.

Custom data types and operators See the following code snippet for definitions of basic data types and operators that we will use to reason about the US Constitution.

There are two data types g and *time* and one derived data type σ. The operators $t\langle op\rangle$ are time dependant versions of operators $\langle op\rangle$.

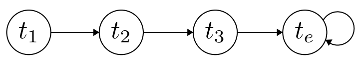
Type g represents the governmental institutions *Congress* , *P(resident)* and *Courts*. The legislative, executive and judicial powers shall later be bestowed upon these three institutions.

There are four instances of time: t_1-t_3 as above and t_e, the instance that marks the end of time. We need t_e to avoid inconsistencies in connection with X. We shall point out where it is necessary when it becomes relevant below.

Since we will only consider a formula's validity at a certain point in time we need time dependant type σ for them, as well as operators lifted to that type, i.e. of type $'a \Rightarrow \sigma$, rather than just $'a \Rightarrow bool$.

Observe that the quantifiers defined may each only be used for one type of argument. This helps with computation times when using tools like Nitpick [1] and Sledgehammer [2] since Isabelle won't have to try different types of arguments.

We also introduce operator X. This requires a precedence relation. To stress the fact that we are talking about a *future* instance of time when using X we call the relation *succ* for successor, rather than *pred* for predecessor. So in Kripke semantics [5] a visualisation of the instances with *succ* as accessibility relation would look as follows:



[4] For example Amend.XVII, the prohibition of intoxicating liquors, was repealed by Amend. XXI, §.1

Based on *succ* we can then define X.
Lastly, we want to define a notion of *validity*. We distinguish between global and local validity.

A formula shall be globally valid when it is valid independently of the current time. This is useful for universally valid definitions such as what we mean by *dictatorship*. A formula shall be locally valid for a specific t if it is valid at that instance of time.

```
datatype g = Congress | P | Courts

datatype time = t1 |t2 |t3 |te

type_synonym σ="(time⇒bool)"

definition tneg :: "σ⇒σ" ("¬_"[52]53)  where "¬φ ≡ λt. ¬φ(t)"
definition tand :: "σ⇒σ⇒σ" (infixr"∧"51) where "φ∧ψ ≡ λt. φ(t)∧ψ(t)"
definition tor  :: "σ⇒σ⇒σ" (infixr"∨"50) where "φ∨ψ ≡ λt. φ(t)∨ψ(t)"
definition timp :: "σ⇒σ⇒σ" (infixr"⟶"49) where "φ⟶ ψ ≡ λt. φ(t)⟶ψ(t)"
definition tequ :: "σ⇒σ⇒σ" (infixr"⟷"48) where "φ⟷ψ ≡ λt. φ(t)⟷ψ(t)"
(**)
definition teq :: "g⇒g⇒σ" (infixr"="40) where "φ=ψ ≡ λt. φ=ψ"
definition tneq :: "g⇒g⇒σ" (infixr"!="40) where "φ!=ψ ≡ λt.¬(φ=ψ)"
(**)
definition tall_g :: "(g⇒σ)⇒σ" ("∀g") where "∀gΦ ≡ λt.∀x. Φ(x)(t)"
definition tallB_g:: "(g⇒σ)⇒σ" (binder"∀g"[8]9) where "∀gx. φ(x) ≡ ∀gφ"
(**)
definition texi_g :: "(g⇒σ)⇒σ" ("∃g") where "∃gΦ ≡ λt.∃x. Φ(x)(t)"
definition texiB_g:: "(g⇒σ)⇒σ" (binder"∃g"[8]9) where "∃gx. φ(x) ≡ ∃gφ"
(**)
definition tall_s :: "(σ⇒σ)⇒σ" ("∀σ") where "∀σΦ ≡ λt.∀φ. Φ(φ)(t)"
definition tallB_s:: "(σ⇒σ)⇒σ" (binder"∀σ"[8]9) where "∀σφ. Φ(φ) ≡ ∀σΦ"
(**)
definition texi_s :: "(σ⇒σ)⇒σ" ("∃σ") where "∃σΦ ≡ λt. ∃φ. Φ(φ)(t)"
definition texiB_s:: "(σ⇒σ)⇒σ" (binder"∃σ"[8]9) where "∃σφ. Φ(φ) ≡ ∃σΦ"

consts  succ::"time⇒time⇒bool"
axiomatization where
  t1_s_t2: "succ t1 t2"      and
  t2_s_t3: "succ t2 t3"      and
  t3_s_te: "succ t3 te"      and
  te_s_te: "succ te te"      and
  Nt1_s_t1: "¬(succ t1 t1)" and
  Nt1_s_t3: "¬(succ t1 t3)" and
  Nt1_s_te: "¬(succ t1 te)" and
  Nt2_s_t1: "¬(succ t2 t1)" and
  Nt2_s_t2: "¬(succ t2 t2)" and
  Nt2_s_te: "¬(succ t2 te)" and
  Nt3_s_t1: "¬(succ t3 t1)" and
  Nt3_s_t2: "¬(succ t3 t2)" and
  Nt3_s_t3: "¬(succ t3 t3)" and
  Nte_s_t1: "¬(succ te t1)" and
  Nte_s_t2: "¬(succ te t2)" and
  Nte_s_t3: "¬(succ te t3)"

definition tnext :: "σ⇒σ" ("X_") where "Xφ ≡ (λt. ∀t'. ((succ t t') ⟶ φ t'))"
definition global_valid :: "σ ⇒ bool" ("⌊_⌋"[7]8) where "⌊φ⌋ ≡ ∀t. φ t"
definition local_valid :: "σ⇒time ⇒ bool" ("⌊_⌋_"[9]10) where "⌊φ⌋t ≡ φ t"
```

Definitions based on the US Constitution Having laid the technical foundations, we provide basic definitions with respect to the Constitution.

We introduce a predicate that expresses whether or not g is a certain branch of government. We require each of the branches to be unique, i.e. each branch has to have a unique governmental institution associated with it. Otherwise, the fact that for example *Congress* is legislative would not imply that P isn't which would make for an unnecessarily complex model.

There is a dictatorship at t if at that instance of time a dictator d exists that represents all branches of government.

```
consts
      is_leg::"g⇒σ"     — ‹g is the legislative›
      is_exe::"g⇒σ"     — ‹g is the executive›
      is_jud::"g⇒σ"     — ‹g is the judiciary›
axiomatization where
      unique_is_leg: "⌊∀⇩g g1. ∀⇩g g2. (((is_leg g1)∧(is_leg g2))⟶(g1 = g2))⌋" and
      unique_is_exe: "⌊∀⇩g g1. ∀⇩g g2. (((is_exe g1)∧(is_exe g2))⟶(g1 = g2))⌋" and
      unique_is_jud: "⌊∀⇩g g1. ∀⇩g g2. (((is_jud g1)∧(is_jud g2))⟶(g1 = g2))⌋"
definition Dictatorship::"σ"
  where "Dictatorship ≡ λt. ∃d. ⌊(is_leg d) ∧ (is_exe d) ∧ (is_jud d)⌋⇩t"
```

Below follow some predicates for formulas $\varphi :: \sigma$. With these we will define properties of the Constitution.

```
consts
      is_amd::"σ⇒σ"        — ‹@{text "φ"} is an amendment›
      is_prop::"σ⇒σ"        — ‹@{text "φ"} is proposed›
      is_rat::"σ⇒σ"          — ‹@{text "φ"} is ratified›
      sup_prop::"g⇒σ⇒σ"  — ‹@{text "φ"} has support by @{text "g"} to be proposed›
      sup_rat::"σ⇒σ"        — ‹@{text "φ"} has support to be ratified›
      maint_suf::"σ⇒σ"    — ‹@{text "φ"} maintains suffrage in Senate for all states›
```

Above predicates help us define the following time dependant properties that will be used in describing the Constitution's state:

```
abbreviation oap::"σ"
  where "oap ≡ ∀⇩σ φ. (¬(is_amd φ))⟶(¬(is_prop φ))"
(**)
abbreviation osp::"σ"
  where "osp ≡ ∀⇩σ φ. ∀⇩g g.(is_leg g)⟶((¬(sup_prop g φ))⟶(¬(is_prop φ)))"
(**)
abbreviation omsp::"σ"
  where "omsp ≡ ∀⇩σ φ. (¬(maint_suf φ))⟶(¬(is_prop φ))"
(**)
abbreviation opr::"σ"
  where "opr ≡ ∀⇩σ φ. (¬(is_prop φ))⟶(¬(X(is_rat φ)))"
(**)
abbreviation osr::"σ"
  where "osr ≡ ∀⇩σ φ. ∀⇩g g. (¬(sup_rat φ))⟶(¬(X(is_rat φ)))"
(**)
abbreviation psr::"σ"
  where "psr ≡ ∀⇩σ φ. (is_prop φ ∧ (sup_rat φ))⟶ (X(is_rat φ))"
(**)
abbreviation rv::"σ"
  where "rv ≡ ∀⇩σ φ. (is_rat φ) ⟶ φ"
```

oap *Only amendments may be proposed.* This time dependant formula is used for technical reasons. It helps to distinguish between generic formulas φ of type σ and what we call amendments. For example *oap* itself may not be proposed if it isn't also declared an amendment.

osp *Only if an amendment has the support of the legislative, can it be proposed.* This is a simplified version of what Art. V says on the amendment process.

omsp *Only amendments that maintain suffrage may be proposed.*

opr *Only proposed amendments may be ratified at the next time instance.*

osr *Only if an amendment has the support for ratification, can it be ratified in the future.*

psr *If an amendment is proposed and has the support for ratification, it will be ratified at the next time instance.* This will be used to show that an amendment proposed at t_i is ratified and thus valid at t_{i+1}, given that it also has support for ratification at t_i. Note that together with *opr* this makes proposition and ratification of an amendment a two-step process.

rv *If an amendment is ratified, it is also valid.* Here the framework for reasoning about amendments is entwined with the content of the amendments. In combination with *psr* this property is a precarious one to work with for, as soon as *rv* is declared to be valid for some t, it will be possible to prove anything as long as it has been proposed with support for ratification in the preceding instance of time.

3 Reasoning with the model

We shall now look into the Constitution's states at instances t_1, t_2 and t_3 by stating axioms and proving properties based on them.

Instance t_1 First we state a few axioms and then give two suggestions of what *amd1* might look like. Observe that all of the properties describing an instance of time, as defined above, are valid at t_1.

```
axiomatization where
    Con_Leg_t1: "⌊is_leg Congress⌋t1" and
    P_Exe_t1: "⌊is_exe P⌋t1" and
    Cou_Jud_t1: "⌊is_jud Courts⌋t1"
axiomatization where
    oap_t1: "⌊oap⌋t1" and
    osp_t1: "⌊osp⌋t1" and
    omsp_t1:"⌊omsp⌋t1" and
    opr_t1: "⌊opr⌋t1" and
    rv_t1:  "⌊rv⌋t1" and
    osr_t1: "⌊osr⌋t1" and
    psr_t1: "⌊psr⌋t1"
definition amd1a::σ
  where "amd1a ≡ ∃σφ. (¬(maint_suf φ))∧((is_prop φ))"
definition amd1b::σ
  where "amd1b ≡ ∀σφ. (is_prop φ)⟶ ((maint_suf φ) ∨ ¬(maint_suf φ))"
```

Neither *amd*1*a* nor *amd*1*b* are optimal solutions. Indeed, there is no optimal solution for the presented framework.

This is because what we want *amd1* to say is that it is not necessary for all proposed amendments to maintain all states' suffrage in Senate. In other words we want condition *omsp* to be omitted at t_2. This, however, is not the same as requiring the amendment to be the negation of *omsp* as *amd1a* does. The negation would require at least one $\varphi :: \sigma$ to expressly *not* maintain suffrage rights for some state *and* be proposed. Yet, it were acceptable both if such a φ existed and if it didn't. We do not want to demand such a φ into existence.

One could therefore choose to use *amd1b* that states a proposed φ may either satisfy the *maint_suf* condition or it may not. Unfortunately, this is a tautology.

Although the suggested amendments do not constitute ideal amendments for the desired outcome, we shall still use them. They help to illustrate how one can reason about amendments within this framework.

Next there are a few axioms that pave the way for the state at t_2. Amendments *amd1a* and *amd1b* are both proposed and have support for ratification at t_1, so they may be ratified at the next instance.

```
axiomatization where
  amd1a_prop_t1:    "⌊is_prop amd1a⌋_t1" and
  amd1a_sup_rat_t1: "⌊sup_rat amd1a⌋_t1" and
  amd1b_prop_t1:    "⌊is_prop amd1b⌋_t1" and
  amd1b_sup_rat_t1: "⌊sup_rat amd1b⌋_t1"

axiomatization where
  XCon_Leg_t1: "⌊X(is_leg Congress)⌋_t1" and
  XP_Exe_t1:   "⌊X(is_exe P)⌋_t1"         and
  XCou_Jud_t1: "⌊X(is_jud Courts)⌋_t1"

axiomatization where
  Xoap_t1:"⌊X oap⌋_t1" and
  Xosp_t1:"⌊X osp⌋_t1" and
  Xopr_t1:"⌊X opr⌋_t1" and
  Xrv_t1: "⌊X rv⌋_t1"  and
  Xosr_t1:"⌊X osr⌋_t1" and
  Xpsr_t1:"⌊X psr⌋_t1"
```

Observe that all Constitution state properties defined above are valid next time, except for *omsp*. This is to ensure that we can introduce an amendment at t_2 that does not satisfy *maint_suf*.

In a way the amendment to Art. V is implemented by simply not using $\lfloor X omsp \rfloor_{t_1}$ as axiom, rather than by working with one of the above suggested amendments *amd1a* and *amd1b*.

Using the axioms provided above, we shall prove that there is no dictatorship at t_1. This requires the proof of facts *only_g_power_t1* meaning that g is the only governmental institution with *(legislative, executive, judicial) power* at t_1. Since g is different for each *power* no dictatorship can be in place at t_1.

```
lemma only_Con_Leg_t1: "⌊∀gg. (is_leg g)⟶(g = Congress)⌋t1"
  unfolding Defs using unique_is_leg Con_Leg_t1
  by (simp add: global_valid_def local_valid_def tallB_g_def tall_g_def tand_def teq_def timp_def)

lemma only_P_Exe_t1:"⌊∀gg. (is_exe g)⟶(g = P)⌋t1"
  unfolding Defs using unique_is_exe P_Exe_t1
  by (simp add: global_valid_def local_valid_def tallB_g_def tall_g_def tand_def teq_def timp_def)

lemma only_Cou_Jud_t1:"⌊∀gg. (is_jud g)⟶(g = Courts)⌋t1"
  unfolding Defs using unique_is_jud Cou_Jud_t1
  by (simp add: global_valid_def local_valid_def tallB_g_def tall_g_def tand_def teq_def timp_def)

theorem noDictatorship_t1: "⌊¬ Dictatorship⌋t1"
  unfolding Defs using only_Con_Leg_t1 only_P_Exe_t1 only_Cou_Jud_t1
  by (metis (no_types, lifting) Dictatorship_def g.distinct(1) local_valid_def tallB_g_def tall_g_de
```

Finally we check whether the axioms so far are even satisfiable by asking Nitpick to find a satisfying model for *True*. Note that we will repeat this test for time instances t_2 and t_3. Since we only ever add axioms and don't remove any, proceeding from one time instance to the next, it is sufficient to only consider the last model provided. We will present this when checking for satisfiability at t_3.

```
lemma T_basic_sat_t1: "True" nitpick[satisfy,user_axioms,show_all,card time = 4]oops
```

Instance t_2 For t_2 we do not need to provide as many axioms as for t_1 since we can deduce $\lfloor\langle property\rangle\rfloor_{t_2}$ from axiom $\lfloor X\langle property\rangle\rfloor_{t_1}$.

```
lemma Con_Leg_t2:"⌊is_leg Congress⌋t2"
  unfolding Defs
  using XCon_Leg_t1 local_valid_def tnext_def t1_s_t2 by auto

lemma P_Exe_t2:"⌊is_exe P⌋t2"
  unfolding Defs using tnext_def XP_Exe_t1
  using XP_Exe_t1 local_valid_def tnext_def t1_s_t2 by auto

lemma Cou_Jud_t2:"⌊is_jud Courts⌋t2"
  using XCou_Jud_t1 local_valid_def tnext_def t1_s_t2 by auto

lemma oap_t2:"⌊oap⌋t2"
  using Xoap_t1 local_valid_def tnext_def t1_s_t2 by auto
lemma osp_t2:"⌊osp⌋t2"
  using Xosp_t1 local_valid_def tnext_def t1_s_t2 by auto
lemma opr_t2:"⌊opr⌋t2"
  using Xopr_t1 local_valid_def tnext_def t1_s_t2 by auto
lemma rv_t2:"⌊rv⌋t2"
  using Xrv_t1 local_valid_def tnext_def t1_s_t2 by auto
lemma osr_t2:"⌊osr⌋t2"
  using Xosr_t1 local_valid_def tnext_def t1_s_t2 by auto
lemma psr_t2:"⌊psr⌋t2"
  using Xpsr_t1 local_valid_def tnext_def t1_s_t2 by auto
```

Below are proofs for the amendments proposed previously. The outline for a validity proof where an amendment *amd* is concerned is as follows:

t_i	t_{i+1}
psr_{t_i}	$rv_{t_{i+1}}$
$\left.\begin{array}{l} is\_prop\ amd \\ sup\_rat\ amd \end{array}\right\} \underset{psr_{t_i}}{\Rightarrow}\ is\_rat\ amd\ \underset{rv_{t_{i+1}}}{\Rightarrow}\ amd$	

This is exactly what we do with *amd1a*.

See below that we can prove $\lfloor amd1b \rfloor_{t_2}$ with or without these axioms since *amd1b* is a tautology. Indeed, we can also show *amd1b*'s validity for t_1 and its global validity. This is not possible with *amd1a*.

```
lemma amd1a_val_t2:"⌊amd1a⌋t2"
proof -
  have "⌊X(is_rat amd1a)⌋t1"
    using amd1a_prop_t1 amd1a_sup_rat_t1 psr_t1 local_valid_def tallB_s_def tall_s_def tand_def timp
    by auto
  thus "⌊amd1a⌋t2"
    using local_valid_def tallB_s_def tall_s_def timp_def tnext_def rv_t2 t1_s_t2
    by auto
qed
lemma amd1b_val_t2:"⌊amd1b⌋t2"
  unfolding Defs
  by (simp add: amd1b_def tallB_s_def tall_s_def timp_def tneg_def tor_def)

lemma amd1b_val_t2_2:"⌊amd1b⌋t2"
  unfolding Defs using amd1b_sup_rat_t1 amd1b_prop_t1 psr_t1 rv_t2
  by (simp add: amd1b_def tallB_s_def tall_s_def timp_def tneg_def tor_def)

lemma amd1b_val_t1:"⌊amd1b⌋t1"
  unfolding Defs
  by (simp add: amd1b_def tallB_s_def tall_s_def timp_def tneg_def tor_def)

lemma amd1b_val:"⌊amd1b⌋"
  unfolding Defs
  by (simp add: amd1b_def tallB_s_def tall_s_def timp_def tneg_def tor_def)
```

Now we introduce *amd2* which will transfer all governmental power to the *President*. Also, we set the stage for t_3 with relevant axioms. As with t_2 we keep all time dependant conditions except for *omsp*.

```
definition amd2::σ where " amd2 ≡ is_leg P ∧ is_exe P ∧ is_jud P"
axiomatization where
  amd2_prop_t2:"⌊is_prop amd2⌋t2" and
  amd2_sup_rat_t2:"⌊sup_rat amd2⌋t2" and
  amd2_not_maint_suf_t2:"⌊¬(maint_suf amd2)⌋t2"

axiomatization where
  Xoap_t2:"⌊X oap⌋t2" and
  Xosp_t2:"⌊X osp⌋t2" and
  Xopr_t2:"⌊X opr⌋t2" and
  Xrv_t2:"⌊X rv⌋t2" and
  Xosr_t2:"⌊X osr⌋t2" and
  Xpsr_t2:"⌊X psr⌋t2"
```

When introducing time instances we mentioned that we needed t_e for technical reasons. This is because we want to use above given axiom $\lfloor X opr \rfloor_{t_2}$ without creating inconsistencies due to a missing successor for t_3.

$$\begin{aligned}
\lfloor X opr \rfloor_{t_2} \Rightarrow \lfloor opr \rfloor_{t_3} & \\
&\Leftrightarrow \lfloor \forall_\sigma \varphi.(\neg(is_prop\ \varphi)) \rightarrow (\neg(X(is_rat\ \varphi))) \rfloor_{t_3} \\
&\Leftrightarrow \lfloor \forall_\sigma \varphi.(X(is_rat\ \varphi)) \rightarrow (is_prop\ \varphi) \rfloor_{t_3} \\
&\Leftrightarrow \forall \varphi.((X(is_rat\ \varphi))t_3) \rightarrow (is_prop\ \varphi)t_3 \\
&\Leftrightarrow \forall \varphi.\forall t'.((succt_3\ t') \rightarrow (is_rat\ \varphi)t') \rightarrow (is_prop\ \varphi)t_3
\end{aligned}$$

If t_3 does not have a successor $(succt_3\ t')$ will always be false, making $(succt_3\ t') \rightarrow (is\_rat\ \varphi)t'$ always true which it shouldn't be. As soon as term $(is\_prop\ \varphi)t_3$ is not true for some φ, axiom $\lfloor X opr \rfloor_{t_2}$ will cause an inconsistency.

We therefore want t_3 to have a successor. In order to avoid circular succession we introduce dummy instance t_e.

Analogously to t_1, we prove properties *only_g_power_t2* to prove *noDictatorship_t2* and check for satisfiability.

```
lemma only_Con_Leg_t2:"⌊∀g g. (is_leg g)⟶(g = Congress)⌋t2"
  using unique_is_leg Con_Leg_t2 global_valid_def local_valid_def tallB_g_def tall_g_def tand_def te
  by simp

lemma only_P_Exe_t2:"⌊∀g g. (is_exe g)⟶(g = P)⌋t2"
  unfolding Defs using unique_is_exe P_Exe_t2
  by (simp add: global_valid_def local_valid_def tallB_g_def tall_g_def tand_def teq_def timp_def)

lemma only_Cou_Jud_t2:"⌊∀g g. (is_jud g)⟶(g = Courts)⌋t2"
  unfolding Defs using unique_is_jud Cou_Jud_t2
  by (simp add: global_valid_def local_valid_def tallB_g_def tall_g_def tand_def teq_def timp_def)

theorem noDictatorship_t2: "⌊¬ Dictatorship⌋t2"
  unfolding Defs using only_Con_Leg_t2 only_P_Exe_t2 only_Cou_Jud_t2 Dictatorship_def
  by (metis (mono_tags, lifting) g.distinct(3) local_valid_def tallB_g_def tall_g_def  tand_def teq_
```

```
lemma T_basic_sat_t2: "True" nitpick[satisfy,user_axioms,show_all,card time = 4]oops
```

Instance t_3 The remainder of this section is rather simple. We prove properties for new time instance t_3 using previously provided axioms *X property_t2*. We then proceed to show that *amd2* is valid with the reasoning given above and use it to prove that there is now a dictatorship.

As before we check that our axioms are satisfiable. For this last instance of time we also give a representation of Nitpick's satisfying model.

```
lemma oap_t3:"⌊oap⌋_t3"
  using Xoap_t2 local_valid_def tnext_def t2_s_t3 by auto
lemma osp_t3:"⌊osp⌋_t3"
  using Xosp_t2 local_valid_def tnext_def t2_s_t3 by auto
lemma opr_t3:"⌊opr⌋_t3"
  using Xopr_t2 local_valid_def tnext_def t2_s_t3 by auto
lemma rv_t3:"⌊rv⌋_t3"
  using Xrv_t2 local_valid_def tnext_def t2_s_t3 by auto
lemma osr_t3:"⌊osr⌋_t3"
  using Xosr_t2 local_valid_def tnext_def t2_s_t3 by auto
lemma psr_t3:"⌊psr⌋_t3"
  using Xpsr_t2 local_valid_def tnext_def t2_s_t3 by auto
```

```
lemma amd2_val_t3:"⌊amd2⌋_t3"
proof -
  have "⌊X(is_rat amd2)⌋_t2"
    using amd2_prop_t2 amd2_sup_rat_t2 local_valid_def tallB_s_def tall_s_def tand_def timp_def tnex
    by auto
  thus  "⌊amd2⌋_t3"
    using local_valid_def tallB_s_def tall_s_def timp_def tnext_def rv_t3 t2_s_t3
    by auto
qed
```

```
theorem Dictatorship_t3:"⌊Dictatorship⌋_t3"
proof -
  have "⌊is_leg P ∧ is_exe P ∧ is_jud P⌋_t3"
    using amd2_val_t3 amd2_def
    by ( simp add: local_valid_def tand_def)
  thus "⌊Dictatorship⌋_t3"
    by (meson Dictatorship_def local_valid_def)
qed
```

```
lemma T_basic_sat_t3: "True" nitpick[satisfy,user_axioms,show_all,format = 2,card time = 4]oops
```

The following satisfying model is the result:[5]

t_1	t_2	t_3	t_e
is_exe P	*is_exe P*	*is_exe P*	*is_exe P*
is_jud Courts	*is_jud Courts*	*is_jud P*	*is_jud P*
is_leg Congress	*is_leg Congress*	*is_leg P*	*is_leg P*
maint_suf amd1a			
sup_prop Congress amd1a			
is_prop amd1a			
sup_rat amd1a	*is_rat amd1a*		
maint_suf amd1b			
sup_prop Congress amd1b			
is_prop amd1b			
sup_rat amd1b	*is_rat amd1b*		
	¬ (maint_suf amd2)		
	sup_prop Congress amd2		
	is_prop amd2		
	sup_rat amd2	*is_rat amd2*	

[5] This is a heavily truncated presentation of the model provided by Nitpick. The successor relation's values are just as defined in "Custom data types and operators".

4 Conclusion

In the course of this work we have explored an argument on how to introduce a dictatorship in the USA without violating the rules laid out in the US Constitution. We did so using proof assistant Isabelle/HOL.

It is an example on how to conduct legal reasoning with the aid of a computer. In this case, there were four main tasks involved: (1) Determining which aspects of the text are relevant. (2) Deciding on a suitable way to represent these concepts in higher order logic. (3) Translating the concepts modelled with HOL to the computer. (4) Conduct reasoning based on the model.

We mainly focused on presenting (3) and (4) in this work since the products of these steps are, by nature, presentable. One can simply provide code. A large part of the benefit of conducting (1) and (2) is finding out what does not work for the text at hand. Presenting findings of this kind would have gone beyond the scope of this paper. That is not to say, however, that they aren't of interest. This brings us to potential further tasks.

The first would be to extend the discussion of a text's modelling to the points that may be considered but that turn out to be unsuitable. This would help others doing similar work. We did this to some degree when explaining about e.g. the necessity of t_e or discussing a suitable representation of *amd1* but many other points could have been mentioned here.

Furthermore, this work only dealt with the contents of the Constitution relevant to the argument formalized. It was a mere case example. In order to conduct general legal reasoning with respect to the US Constitution it is necessary to analyse and represent more of its contents, rather than just one small part.

Lastly, when it comes to formalizing legal concepts in general the collaboration of logicians and legal scholars is essential to achieve better results. Given that the problems presented above are in nature interdisciplinary they should also be solved in an interdisciplinary context.

References

1. Blanchette, J.C.: Picking Nits–A user's guide to nitpick for Isabelle/HOL (2019), http://isabelle.in.tum.de/dist/doc/nitpick.pdf
2. Blanchette, J.C., Paulson, L.C.: Hammering away– A user's guide to Sledgehammer for Isabelle/HOL (2019), https://isabelle.in.tum.de/dist/doc/sledgehammer.pdf
3. Dawson, J.W.: Logical dilemmas: The life and work of Kurt Gödel. AK Peters/CRC Press (1997)
4. Feferman, S., Dawson, J.W.J., Goldfarb, W., Parsons, C., Sieg, W. (eds.): Collected Works, vol. 5. Oxford University Press ; Clarendon Press (2003)
5. Garson, J.: Modal logic. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, fall 2018 edn. (2018)
6. Gödel, K.: Briefe an marianne gödel 1906-1978. https://www.digital.wienbibliothek.at/wbr/nav/classification/2559756 (1906-1978), accessed: 2019-08-20
7. Goranko, V., Galton, A.: Temporal logic. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, winter 2015 edn. (2015)
8. Guerra-Pujol, F.: Gödel's loophole. Cap. UL Rev. **41**, 637 (2013)
9. Levinson, S., Balkin, J.M.: Constitutional dictatorship: Its dangers and its design. Minn. L. Rev. **94**, 1789 (2009)
10. Morgenstern, O.: Oskar morgenstern's account of kurt gödel's naturalization. Dorothy Morgenstern Thomas collection (1971)
11. Morgenstern, O.: Tagebuch. digitale edition: 1917 bis 1977. http://gams.uni-graz.at/archive/objects/o:ome.b47-47/methods/sdef:TEI/get?mode=b47-47 (2016), accessed: 2019-08-07
12. Wang, H.: Reflections on Kurt Gödel. MIT Press (1987)
13. Wenzel, M.: The Isabelle/Isar Reference Manual (2019), https://isabelle.in.tum.de/doc/isar-ref.pdf
14. Yourgrau, P.: A world without time: The forgotten legacy of Gödel and Einstein. Basic Books (2006)