Type Definitions

Example 10

Title: Type Definitions
SML files: none
Objective: Defining Data Types and Proving Their Properties

1 Introduction

Defining types is a means to introduce new data structures with associated properties. The new data structures allow us to use new forms or syntax. The properties associated with the new forms allow us to reason about them.

The means by which new types are introduced is the method described by Melham [2] and is used in the Higher Order Logic theorem-prover, [1].

1. An appropriate subset of an existing type is identified to represent the new type.

2. The new type is introduced into the logic by means of an inference rule which yields a type definition theorem containing the new type.

3. The properties of the new type are derived from the properties of the underlying representation.

The rest of this paper is organized as follows. Section 2 describes two inference rules used in type definition. Section 3 introduces representation and abstraction functions in a standard way with proofs of their properties. Section 4 goes through a simple example in detail.

2 Inference Rules

2.1 Select Rule

A new operator $\varepsilon$ is used to denote values which satisfy a property $P$. Its semantics is given by the following rule.

$$
\frac{\exists x . P(x)}{P(\varepsilon x . P(x))} \text{ Select \textendash Rule}
$$

(1)
What the rule says is if there is at least one value \( x \) which has property \( P \), then \( \varepsilon x.P(x) \) is a value which has property \( P \). For example, if \( P(x) = 0 \leq x \leq 3 \), then \( \varepsilon x.P(x) \) denotes any one of the values in the set \{0, 1, 2, 3\}. Which particular value in the set is not specified. If no value satisfies \( P \), then \( \varepsilon x.P(x) \) denotes some arbitrary value.

Why are we interested in the \( \varepsilon \) operator? Many times, all we really care about are the properties a value may have as opposed to one particular value. For example, if we only need to refer to an even number where any even number will do, we use \( \varepsilon x.\text{Even}(x) \). Later on, we will want to refer to functions which are isomorphic where any isomorphic function will do. In this case, we represent any such function as \( \varepsilon f.\text{isomorphic}(f) \).

### 2.2 Representing New Types

New types are defined in terms of subsets of existing types. An isomorphic or one-to-one and onto mapping between elements of the new type and elements in the subset of the existing type is established. This mapping is the representation function which defines elements of the new type in terms of elements of the existing type. The properties of the new type are precisely those properties of the elements in the underlying representation.

The subset of the existing type is identified by a predicate \( P \). Typically, there are many possible mappings between elements of the new type and elements in the subset of the existing type. The only requirement is those mappings be isomorphic.

A proper representation function \( \text{rep} \) for a given \( P \) satisfies the \( \text{TYPE.DEF} \) predicate defined below.

\[
\text{TYPE.DEF} \ P \ \text{rep} = \quad (2)
\]

\[
(\forall a_1 \ a_2. \text{rep} \ a_1 = \text{rep} \ a_2 \supset a_1 = a_2) \land (\forall r. P \ r \equiv \exists a. r = \text{rep} \ a)
\]

where \( a, a_1, \) and \( a_2 \) are elements of the new type and \( r \) is a member of the existing type used to represent the new type. \( \forall a_1 \ a_2. \text{rep} \ a_1 = \text{rep} \ a_2 \supset a_1 = a_2 \) is the “one-to-one” property. It says that if the representations of \( a_1 \) and \( a_2 \) are the same, then \( a_1 = a_2 \). \( \forall r. P \ r \equiv \exists a. r = \text{rep} \ a \) is the “onto” property. It says every \( r \) in the subset \( P \) is a representation of some element \( a \) in the new type.

### 2.3 New Type Definition

If a predicate \( P \) identifies a non-empty subset of an existing type, it can be used to represent a new type. The \text{New Type} inference rule does this.

\[
\frac{\exists x. P(x)}{\exists \text{rep}. \text{TYPE.DEF} \ P \ \text{rep}} \quad \text{New Type}
\]

(3)

Used in conjunction with Select Rule, we can refer to any representation function of a new type which we know is an isomorphic function between elements of the new type and elements of an existing type satisfying \( P \).

### 3 Standard Properties

Once we have introduced a new type \( ty_P \) in terms of an existing type \( ty \), we can introduce representation and abstraction functions \( REP \) and \( ABS \) between \( ty_P \) and \( ty \) and prove their
properties in a standard way. In fact, this is done automatically in theorem-provers such as HOL.

A representation function which maps elements of new type \( ty_p \) to elements of an existing type \( ty \) is *any* function \( rep : ty_p \to ty \) that is isomorphic. We define representation \( REP \) to be any such function as follows.

\[
REP = \varepsilon_{rep\cdot\text{TYPE}\_\text{DEF}} P \ rep
\]

In a similar fashion, we define the abstraction function \( ABS : ty \to ty_p \) in terms of \( REP \). \( ABS \ r \) returns the element \( a : ty_p \) whose representation is \( r \).

\[
ABS \ r = \varepsilon a.r = \text{REP} \ a
\]

Given the above definitions, the following *standard* properties are provable.

\[
\forall a_1 a_2. \text{REP} \ a_1 = \text{REP} \ a_2 \supset a_1 = a_2 \tag{6}
\]

\[
\forall r. P \ r \equiv (\exists a.r = \text{REP} \ a) \tag{7}
\]

\[
\forall r_1 r_2. P \ r_1 \supset P \ r_2 \supset (ABS \ r_1 = ABS \ r_2 \supset r_1 = r_2) \tag{8}
\]

\[
\forall a. ABS(\text{REP} \ a) = a \tag{9}
\]

\[
\forall r. P \ r \equiv (\text{REP}(ABS \ r) = r) \tag{10}
\]

\[
\forall a. \exists r.(a = ABS \ r) \wedge P \ r] \tag{11}
\]

The proofs of each of the above properties are presented in the following sections.

### 3.1 REP is Isomorphic

Proving \( REP \) is isomorphic deals with the first two standard properties. We will take for granted the definitions of \( \text{TYPE}\_\text{DEF}, \text{REP}, \) and \( ABS \) and not include them in the list of assumptions for brevity.

The only assumption we will start with is the property that \( P \) is non-empty, i.e. \( \exists x.P \ x \) is true. The proof that \( REP \) is isomorphic appeals directly to the fact that we defined \( REP \) as any function satisfying \( \text{TYPE}\_\text{DEF} \) which itself defines the property of being isomorphic.

---

| A0. \( x = x \) |   |
| A1. \( \exists x. P \ x \) |   |
| \( G1. (\forall a_1 a_2. \text{REP} \ a_1 = \text{REP} \ a_2 \supset a_1 = a_2) \wedge (\forall r. P \ r \equiv (\exists a.r = \text{REP} \ a)) \) |   |
| A2. By A1 – New – Type |   |
| \( \exists_{\text{rep}\cdot\text{TYPE}\_\text{DEF}} P \ \text{rep} \) |   |
| A3. By A2 – Select |   |
| \( \text{TYPE}\_\text{DEF} \ P \ (\text{rep}\cdot\text{TYPE}\_\text{DEF} \ P \ \text{rep}) \) |   |
| A4. By A3 REP definition |   |
| \( \text{TYPE}\_\text{DEF} \ P \ \text{REP} \) |   |
| A5. By A4 TYPE\_DEF definition |   |
| \((\forall a_1 a_2. \text{REP} \ a_1 = \text{REP} \ a_2 \supset a_1 = a_2) \wedge (\forall r. P \ r \equiv (\exists a.r = \text{REP} \ a)) \) |   |
| G2. By A0, A5 – resolution |   |
| \( \text{true} \) |   |
3.2 ABS is One-to-One

If \( r_1 \) and \( r_2 \) are in the subset defined by \( P \), then \( \text{ABS} \ r_1 = \text{ABS} \ r_2 \) implies \( r_1 = r_2 \). We use the isomorphic property of \( \text{REP} \) to prove this.

\[
\begin{align*}
A0. \ x &= x \\
A1. \ \exists x. P \ x \\
A2. \ \text{REP} \ a_1 &= \text{REP} \ a_2 \supset a_1 = a_2 \\
A3. \ P \ r \equiv (\exists a. x = \text{REP} \ a) \\
\quad & G1. \ \forall r_1 \ r_2, P \ r_1 \supset P \ r_2 \supset (\text{ABS} \ r_1 = \text{ABS} \ r_2 \supset r_1 = r_2) \\
& \text{By } \forall -\text{elimination and repeated if } \exists -\text{splitting} \\
A4. \ P \ r_1' \\
A5. \ P \ r_2' \\
A6. \ \text{ABS} \ r_1' = \text{ABS} \ r_2' \\
\quad & G2. \ r_1' = r_2' \\
A7. \ \text{By A4, A3 EQ } \rightarrow \text{LR} \\
\quad & \exists a. r_1' = \text{REP} \ a \\
A8. \ \text{By A7 } \forall -\text{elim} \\
\quad & r_1' = \text{REP} \ a_1' \\
A9. \ \text{By A5, A3 EQ } \rightarrow \text{LR} \\
\quad & \exists a. r_2' = \text{REP} \ a \\
A10. \ \text{By A9 } \forall -\text{elim} \\
\quad & r_2' = \text{REP} \ a_2' \\
A11. \ \text{By A7 Select} \\
\quad & r_1' = \text{REP}(\exists a. r_1' = \text{REP} \ a) \\
A12. \ \text{By A9 Select} \\
\quad & r_2' = \text{REP}(\exists a. r_2' = \text{REP} \ a) \\
A13. \ \text{By A11 ABS definition} \\
\quad & r_1' = \text{REP}(\text{ABS} \ r_1') \\
A14. \ \text{By A12 ABS definition} \\
\quad & r_2' = \text{REP}(\text{ABS} \ r_2') \\
A15. \ \text{By A13 A6 EQ } \rightarrow \text{LR} \\
\quad & r_1' = \text{REP}(\text{ABS} \ r_1') \\
A16. \ \text{By A15 A14 EQ } \rightarrow \text{LR} \\
\quad & r_1' = r_2' \\
G3. \ \text{By A16 G2 resolution} \\
\quad & \text{true}
\end{align*}
\]

3.3 ABS is Left Inverse of REP

For any member of the new type, \( \text{ABS} \circ \text{REP} \) is the identity function. The goal is to prove \( \forall a. \text{ABS}(\text{REP} \ a) = a \). By quantifier elimination and Skolemization, the equivalent goal to prove is \( \text{ABS}(\text{REP} \ a') = a' \) for a Skolem constant \( a' \). It is easy to prove \( \exists a. \text{REP} \ a' = \text{REP} \ a \) by quantifier elimination and unification. Using this fact with the Select-Rule we can infer \( \text{REP} \ a' = \text{REP}(\exists a. \text{REP} \ a' = \text{REP} \ a) \). \( \exists a. \text{REP} \ a' = \text{REP} \ a \) is exactly the definition of \( \text{ABS}(\text{REP} \ a') \). From the one-to-one property of \( \text{REP} \) as given in A2 below, we conclude that \( a' = \text{ABS}(\text{REP} \ a') \).
3.4 REP is the Left Inverse of ABS

When \( r \) is in the subset defined by \( P \) then \( REP \circ ABS \) is the identity function. The goal is to prove \( \forall r. P r \equiv (REP(ABS r) = r) \). By quantifier elimination, Skolemization, and case analysis, the two subgoals which must be proved are: 1) \( P r' \ni (REP(ABS r') = r') \) and 2) \( REP(ABS r') = r' \ni P r' \).

The first subgoal is proved using the property \( P r \equiv (\exists a. r = REP a) \) which is A3 below. From \( P r' \) we infer \( \exists a. r = REP a \). Using the Select-Rule we get \( r' = REP(\exists a. r' = REP a) \). From the definition of \( ABS \), this yields \( r' = REP(ABS r') \).

The second subgoal is to prove \( P r' \) under the assumption \( REP(ABS r') = r' \). By A3, this is equivalent to proving \( \exists a. r' = REP a \). By \( \exists \)-elimination, this is equivalent to proving \( r' = REP a \) where \( a \) is a free variable. Using the assumption \( r' = REP(ABS r') \), the goal reduces to finding a variable assignment for \( a \) such that \( REP(ABS r') = REP a \). The proof is completed by unifying \( a \) with \( ABS r' \).
A0. $x = x$
A1. $\exists x. P \ x$
A2. $\text{REP } a_1 = \text{REP } a_2 \supset a_1 = a_2$
A3. $P \ r \equiv (\exists a. x = \text{REP } a)$
A4. $P \ r_1 \supset P \ r_2 \supset (\text{ABS } r_1 = \text{ABS } r_2 \supset r_1 = r_2)$
A5. $\text{ABS}(\text{REP } a) = a$
   
   G1. $\forall r. P \ r \equiv (\text{REP}(\text{ABS } r) = r)$
   G2. By G1 $\forall - \text{ elim}$
   $P \ r' \equiv (\text{REP}(\text{ABS } r') = r')$

Case analysis on biconditional goal
G3.1 $P \ r' \supset (\text{REP}(\text{ABS } r') = r')$

By if $-$ split
A6.1 $P \ r'$
   
   G4.1 $\text{REP}(\text{ABS } r') = r'$
A7.1 By A6.1 A3 EQ $-$ LR
   $\exists a. r' = \text{REP } a$
A8.1 By A7.1 Select $-$ Rule
   $r' = \text{REP}(\varepsilon a. r' = \text{REP } a)$
A9.1 By A8.1 ABS definition
   $r' = \text{REP}(\text{ABS } r')$

By A9.1 G4.1 resolution
   $\text{true}$

G3.2 $\text{REP}(\text{ABS } r') = r' \supset P \ r'$
By G3.2 if $-$ split
A6.2 $\text{REP}(\text{ABS } r') = r'$
   
   G4.2 $P \ r'$
   
   G5.2 By G4.2 A3 EQU $-$ LR
   $\exists a. r' = \text{REP } a$
   G6.2 By G5.2 $\exists - \text{ elim}$
   $r' = \text{REP } a$
   G7.2 By G6.2 A6.2 EQ $-$ RL
   $\text{REP}(\text{ABS } r') = \text{REP } a$
   G8.2 By A0 G7.2 resolution $\theta = \{a \leftarrow \text{ABS } r'\}$

   $\text{true}$

3.5 ABS is Onto

For every $a$ in the new type there is a corresponding member $r$ of the existing type which is in the subset $P$. Tje goal is to prove $\forall a. \exists r. [(a = \text{ABS } r) \land P \ r]$. By quantifier elimination and Skolemization, the equivalent goal to prove is $(a' = \text{ABS } r) \land P \ r$. We again use the (easily proved) fact that $\exists a. \text{REP } a' = \text{REP } a$. From the Select-Rule we infer $\text{REP } a' = \text{REP}(\varepsilon a. \text{REP } a' = \text{REP } a)$. Using the definition of ABS we get $\text{REP } a' = \text{REP}(\text{ABS}(\text{REP } a'))$. From the one-to-one properties of REP as found in A2 below, we infer $a' = \text{ABS}(\text{REP } a')$. We simplify the goal using this fact to $P(\text{REP } a')$. Using A3, the goal becomes $\exists a. \text{REP } a' = \text{REP } a$ which we know to be true.

CSE 607: Types: 6
A0. \( x = x \)
A1. \( \exists x. P \ x \)
A2. \( \text{REP} \ a_1 = \text{REP} \ a_2 \supset a_1 = a_2 \)
A3. \( P \ r \equiv (\exists a. x = \text{REP} \ a) \)
A4. \( P \ r_1 \supset P \ r_2 \supset (\text{ABS} \ r_1 = \text{ABS} \ r_2 \supset r_1 = r_2) \)
A5. \( \text{ABS}(\text{REP} \ a) = a \)
A6. \( P \ r \equiv (\text{REP}(\text{ABS} \ r) = r) \)
G1. \( \forall a. \exists r.([a = \text{ABS} \ r] \land P \ r) \)
G2. By G1 \( \forall - \) elim
\( \exists r.([a' = \text{ABS} \ r] \land P \ r) \)
G3. By G2 \( \exists - \) elim
\( (a' = \text{ABS} \ r) \land P \ r \)
A7. The following is easy to prove
\( \exists a. \text{REP} \ a' = \text{REP} \ a \)
A8. By A7 Select
\( \text{REP} \ a' = \text{REP}(\exists a. \text{REP} \ a' = \text{REP} \ a) \)
A9. By A8 ABS definition
\( \text{REP} \ a' = \text{REP}(\text{ABS}(\text{REP} \ a')) \)
A10. By A9 A2 resolution
\( a' = \text{ABS}(\text{REP} \ a') \)
G4. By A10 G3 resolution \( \theta = \{ r \leftarrow \text{REP} \ a' \} \)
\( P(\text{REP} \ a') \)
G5. A3 G4 EQU – LR
\( \exists a. \text{REP} \ a' = \text{REP} \ a \)
G6. By A7 G5 resolution
\( \text{true} \)

4 Color Example

In this section we go through a detailed development of how the data type \( BW \) is introduced. In Backus-Naur Form (BNF) we would define \( BW \) as follows.

\[
BW ::= \text{Black} \mid \text{White}
\]  

(12)

The properties we would hope to prove are:

\( \forall (x : BW).(x = \text{Black}) \lor (x = \text{White}) \)  

(13)

\( \text{Black} \neq \text{White} \)  

(14)

\( P(\text{Black}) \land P(\text{White}) \supset \forall (x : BW).P(x) \)  

(15)

The proofs of these properties depends entirely on the underlying representation of \text{Black} and \text{White}.

Figure 1 shows the relationships between the type \( BW \) and its underlying representation in the type \text{bool} × \text{bool}. From the figure we see that \text{Black} is represented by \( (T, F) \) while \text{White} is represented by \( (F, T) \). The predicate \( \text{isBW} \) identifies the (non-empty) subset of \text{bool} × \text{bool} which is used to represent the elements of \( BW \). The function \( \text{REP}_BW \) maps elements of \( BW \) into their representations. The function \( \text{ABS}_BW \) maps elements of \text{bool} × \text{bool} which satisfy \( \text{isBW} \) to their corresponding elements in \( BW \). Both \( \text{REP}_BW \) and \( \text{ABS}_BW \) are isomorphic and are inverses of one another.
4.1 Identify Subset

The subset of the existing type `bool x bool` used to represent BW is defined by `isBW`. `isBW` \( x \) is true if \( x = (T, F) \) or \( x = (F, T) \).

\[
isBW \ x = [(x = (T, F)) \lor (x = (F, T))]
\]  

(16)

After `isBW` is defined, we prove that there is at least one `x` which satisfies `isBW`. Doing so shows that `isBW` defines a non-empty subset of `bool x bool` which can be used to define `BW`.

\[
\begin{align*}
A0. \ x &= x \\
A1. \ isBW \ x &= [(x = (T, F)) \lor (x = (F, T))] \\
G1. \ \exists x. isBW \ x \\
G2. \ By \ G1 \ \exists x. isBW \ x \\
G3. \ By \ G2 \ \exists x. isBW \ x \\
G4. \ By \ A0 \ G3 \ \text{resolution}
\end{align*}
\]

4.2 Introduce the Type

The type BW is introduced using `TYPE_DEF`. `TYPE_DEF` is a parameterized predicate with parameters `P` and `rep`. `P` is the predicate which identifies a (non-empty) subset of an existing type `typ` used to represent the new type `ty`. `rep` is a function of type `ty -> typ` which is isomorphic, that is one-to-one and onto with respect to the elements of type `ty` and the elements of `typ` satisfying `P`.

\[
\begin{align*}
\text{TYPE_DEF} \ P \ rep &= \ (\forall a_1 a_2. rep a_1 = rep a_2 \Rightarrow a_1 = a_2) \land \ (\forall r. P r \equiv \exists a. r = rep a) \\
\end{align*}
\]

(17)

Using the `New Type` rule and `\( \exists x. isBW \ x \)` we can conclude that there is a `rep` which satisfies `TYPE_DEF isBW rep`.

\[
\exists rep. \text{TYPE_DEF isBW rep}
\]

(18)
4.3 Define the REP_BW and ABS_BW Functions

The representation and abstraction functions REP_BW and ABS_BW are defined using the \( \varepsilon \) operator in the standard fashion. REP_BW is any representation function satisfying TYPE_DEF, i.e. any isomorphic function from BW to \( \{(T,F),(F,T)\} \). ABS_BW \( r \) is the function which returns \( a \in BW \) such that \( r = REP_BW a \).

\[
\begin{align*}
REP_BW &= \varepsilon \text{rep.TYPE_DEF isBW rep} \\
ABS_BW r &= \varepsilon a.r = REP_BW a
\end{align*}
\]

4.4 Prove Standard Properties of ABS_BW and REP_BW

The standard properties of ABS_BW and REP_BW are proved as previously shown in Section 3.

\[
\begin{align*}
REP_BW a_1 &= REP_BW a_1 \supset a_1 = a_2 \\
isBW r_1 \supset isBW r_2 \supset (ABS_BW r_1 &= ABS_BW r_2 \supset r_1 = r_2) \\
ABS_BW(REP_BW a) &= a \\
isBW r &\equiv (REP_BW(ABS_BW r) = r) \\
\exists r.(a = ABS_BW r) \land isBW r
\end{align*}
\]
4.5 Define Black and White

Finally, we are able to define Black and White in terms of their representations \((T, F)\) and \((F, T)\).

\[
\begin{align*}
\text{Black} & = \text{ABS}_\text{BW}(T, F) \\
\text{White} & = \text{ABS}_\text{BW}(F, T)
\end{align*}
\] (26) (27)

4.6 Proof of Cases Theorem

Based on the definition of Black and White, and on the properties of \text{ABS}_\text{BW} and \text{REP}_\text{BW} we can prove the cases theorem for the type BW.

\[
\forall x : \text{BW}. (x = \text{Black}) \lor (x = \text{White})
\] (28)

The proof is straightforward. First, the definitions of Black and White in terms of \text{ABS}_\text{BW} are used to expand the goal. Using the property that:

\[
\exists r. (a = \text{ABS}_\text{BW} r) \land \text{isBW} r
\] (29)

and the definition of isBW:

\[
\text{isBW} x = [(x = (T, F)) \lor (x = (F, T))]
\] (30)
we can derive the relationship:

\[ r' = (T, F) \lor r' = (F, T) \] (31)

This allows us to do a case analysis on \( r' \).

\begin{verbatim}
A0. \( x = x \)
A1. \( isBW \ x = [(x = (T, F)) \lor (x = (F, T))] \)
A2. \( \exists x. isBW x \)
A3. \( TYPE_DEF \ P \ rep = (\forall a_{1}a_{2}. rep\ a_{1} = rep\ a_{2} \supset a_{1} = a_{2}) \land (\forall r. r \equiv \exists a. r = rep\ a) \)
A4. \( \exists rep. TYPE_DEF \ isBW \ rep \)
A5. \( REP_BW = \epsilon rep. TYPE_DEF \ isBW \ rep \)
A6. \( ABS_BW \ r = \epsilon a. r = REP_BW \ a \)
A7. \( REP_BW \ a_{1} = REP_BW \ a_{1} \supset a_{1} = a_{2} \)
A8. \( isBW \ r_{1} \supset isBW \ r_{2} \supset (ABS_BW \ r_{1} = ABS_BW \ r_{2} \supset r_{1} = r_{2}) \)
A9. \( ABS_BW(REP_BW \ a) = a \)
A10. \( isBW \ r \equiv (REP_BW(ABS_BW \ r) = r) \)
A11. \( \exists r. (a = ABS_BW \ r) \land isBW \ r \)
A12. \( Black = ABS_BW(T, F) \)
A13. \( White = ABS_BW(F, T) \)
   G1. \( \forall x : BW. (x = Black) \lor (x = White) \)
   G2. By G1 \( \forall - \) elim
       \( (x' = Black) \lor (x' = White) \)
   G3. By G2 A12 A13EQ - LR
       \( (x' = ABS_BW(T, F)) \lor (x' = ABS_BW(F, T)) \)
A14. By A11 \( \theta = \{a \leftarrow x'\} \) and \( \forall - \) elim
       \( (x' = ABS_BW \ r') \land (isBW \ r') \)
A15. By A14 A1 EQ - LR
       \( r' = (T, F) \lor r' = (F, T) \)
\end{verbatim}

Case 1
A16.1 \( r' = (T, F) \)
A17.1 By A14 A16.1 EQ - LR
   \( x' = ABS_BW(T, F) \)
   G4.1 By A17.1 G3 resolution
   \( true \)

Case 2
A16.2 \( r' = (F, T) \)
A17.2 By A14 A16.2 EQ - LR
   \( x' = ABS_BW(F, T) \)
   G4.2 By A17.2 G3 resolution
   \( true \)

4.7 Proof of Distinctiveness Theorem

The proof of \( Black \neq White \) is done by creating a contradiction. The key property is A8: \( isBW \ r_{1} \supset isBW \ r_{2} \supset (ABS_BW \ r_{1} = ABS_BW \ r_{2} \supset r_{1} = r_{2}) \). By duality and the definitions of \( Black \) and \( White \), we get \( ABS_BW(T, F) = ABS_BW(F, T) \). From the definition of \( isBW \) we know \( isBW(T, F) \) and \( isBW(F, T) \) are true. All the above used with A8 produces the contradiction \( (T, F) = (F, T) \).
A0. $x = x$
A1. $isBW\ x = [(x = (T,F)) \lor (x = (F,T))]$
A2. $\exists x. isBW\ x$
A3. $TYPE.DEF\ P\ rep = (\forall a_1\ a_2. rep\ a_1 = rep\ a_2 \supset a_1 = a_2) \land (\forall r. P\ r \equiv \exists a. r = rep\ a)$
A4. $\exists rep. TYPE.DEF\ isBW\ rep$
A5. $REP.BW = \varepsilon\ rep. TYPE.DEF\ isBW\ rep$
A6. $ABS.BW\ r = \varepsilon\ a. r = REP.BW\ a\ A7. REP.BW\ a_1 = REP.BW\ a_1 \supset a_1 = a_2$
A8. $isBW\ r_1 \supset isBW\ r_2 \supset (ABS.BW\ r_1 = ABS.BW\ r_2 \supset r_1 = r_2)$
A9. $ABS.BW(REP.BW\ a) = a$
A10. $isBW\ r \equiv (REP.BW(ABS.BW\ r) = r)$
A11. $\exists r. (a = ABS.BW\ r) \land isBW\ r$
A12. $Black = ABS.BW(T,F)$
A13. $White = ABS.BW(F,T)$
A14. $(x = Black) \lor (x = White)$
   \hspace{1cm} G1. $\neg(Black = White)$
   \hspace{1cm} G2. By G1 A12 A13 EQ - LR
   \hspace{1cm} G3. $\neg(ABS.BW(T,F) = ABS.BW(F,T))$
A15. By G3 Duality
   $ABS.BW(T,F) = ABS.BW(F,T)$
A16. By A1 $\theta = \{x \leftarrow (T,F)\}$
   $isBW(T,F)$
A17. By A1 $\theta = \{x \leftarrow (F,T)\}$
   $isBW(F,T)$
A18. By A8, A16, A17, A15 resolution
   $(T,F) = (F,T)$
   $false$

### 4.8 Proof of Induction Theorem

The proof of the induction theorem $P(White) \land P(Black) \supset \forall x. P(x)$ follows easily from the cases property A14: $(x = Black) \lor (x = White)$. The goal $P(White) \land P(Black) \supset \forall x. P(x)$ is analyzed for each case, i.e. for the case when $x = Black$ and $x = White$. 
A0. \( x = x \)
A1. isBW \( x = [(x = (T, F)) \lor (x = (F, T))] \)
A2. \( \exists x. isBW x \)
A3. TYPE.DEF \( P \) rep = \( (\forall a_1 a_2. rep a_1 = rep a_2 \supset a_1 = a_2) \land (\forall r. r \equiv \exists a. r = rep a) \)
A4. \( \exists x. TYPE.DEF isBW \) rep
A5. REP_BW = \( \epsilon_{\text{rep}. TYPE.DEF} isBW \) rep
A6. ABS_BW \( r = \varepsilon a. r = REP_BW a \)
A7. REP_BW \( a_1 = REP_BW a_1 \supset a_1 = a_2 \)
A8. isBW \( r_1 \supset isBW r_2 \supset (ABS_BW r_1 = ABS_BW r_2 \supset r_1 = r_2) \)
A9. ABS_BW(\( REP_BW a \)) = a
A10. isBW \( r \equiv (REP_BW(ABS_BW r) = r) \)
A11. \( \exists r. (a = ABS_BW r) \land isBW r \)
A12. Black = ABS_BW(T, F)
A13. White = ABS_BW(F, T)
A14. \( (x = \text{Black}) \lor (x = \text{White}) \)
A15. \( \neg (\text{Black} = \text{White}) \)

By G1 if \(-\) split
A16. \( P(\text{White}) \land P(\text{Black}) \)

G2. \( \forall x. P(x) \)
G3. By G2 \(-\) elim

\( P(x') \)
A17. By A14 \( \theta = \{x \leftarrow x'\} \)

\( x' = \text{Black} \lor x' = \text{White} \)

Case 1
A18.1 \( x' = \text{Black} \)

G4.1 By G3 A18.1 EQ – LR

\( P(\text{Black}) \)
G5.1 By A16 G4.1 resolution

\( \text{true} \)

Case 2
A18.2 \( x' = \text{White} \)

G4.2 By G3 A18.2 EQ – LR

\( P(\text{White}) \)
G5.2 By A16 G4.2 resolution

\( \text{true} \)

References
