# Baby SNARKs

Ashvni Narayanan, for Yatima Inc

July 13, 2022*

This file describes the implementation of the soundness proof of the Baby SNARKs program. The aim is to explain the code of the proof of soundness in [1]. The mathematics is given in [2], and the code shall be explained in the same notation.

# 1 Code

## 1.1 Setup

Some notes :

- The `open_locale big_operators` command lets us use the local notation for sums ($\sum$) and products ($\prod$), as defined in the file [3].
- By declaring `universes u`, one assumes that all elements have `Type u`.
- `parameters` is the same as `variables`, and is used to declare variables that have scope in a given `section`. In this case, they are valid throughout the file.
- It would help to `open polynomial` (open the namespace `polynomial`) at the beginning of the file, one then does not need to add the prefix to each lemma that is called from that namespace.

We have as variables `F`, which is a field (although it is mentioned that this is the finite field parameter of the SNARK, the finiteness is nowhere stated or used). We also have the natural number variables `m`, `n_stmt` and `n_wit`. These are $m$, $l$ and $n-l$ in [2]. $n$ is defined to be the sum of `n_stmt` and `n_wit`.

The collection of polynomials $u_0, u_1, \cdots u_{l-1}$ are defined here. The author defines it in terms of a function `u_stmt`, which takes an element of $\mathbb{Z}/l\mathbb{Z}$ and returns a polynomial with $F$-coefficients. Note that `fin n_stmt` is nothing but the set of natural numbers up to $l$, or equivalently, $\mathbb{Z}/l\mathbb{Z}$. `u_wit` is defined similarly to denote the polynomials $u_l, u_{l+1}, \cdots u_{n-1}$. The roots of the polynomial $t$ are defined in the same fashion, with `r i` denoting $r_i$, for $0 \leq i \leq m-1$.

The polynomial $t$ is then defined as $t = \prod_{i=0}^{m-1}(X - r_i)$ here. `polynomial.X` denotes $X$ as a polynomial in $F[X]$, and `polynomial.C (r i)` denotes the constant polynomial $r_i$.

## 1.2 Properties of $t$

The lemma `nat_degree_t` says :

**Lemma 1.** *The degree of $t$ is $m$.*

`nat_degree` returns the degree of the polynomial as a natural number. This differs from `polynomial.degree` only when the polynomial is zero. The proof follows simply by noting that the degree of the product of the polynomials $\prod_{i=0}^{m-1}(X - r_i)$ is the sum of the degrees of $X - r_i$ (`nat_degree_prod`), as long as each of these are nonzero (`X_sub_C_ne_zero`).

The lemma `monic_t` then says :

*This document may be updated frequently.

**Lemma 2.** *The polynomial $t$ is monic.*

The proof follows from the fact that a product of monic polynomials is monic (`monic_prod_of_monic`), and that each $(X - r_i)$ is monic (`monic_X_sub_C`).

The next lemma `degree_t_pos` tells us :

**Lemma 3.** *If $0 < m$, then the degree of $t$ is positive.*

Note that this lemma uses `degree` instead of `nat_degree`. As a result, we must prove that $m$ is nonzero implies $t$ being nonzero, in which case `nat_degree` and `degree` coincide.

Before getting into the proof, let us first understand the reason for the distinction between `nat_degree` and `degree`. Lean uses the inductive type `option`. Basically, given `A`, `option A` comprises of `none` (the undefined element) and `some a` for all elements `a` of `A`. The function `option.get_or_else a` returns `b` when given `some b` and `a` when given `none`. Given a polynomial `p`, `degree p` returns `some` of the supremum of all numbers $n$ such that $X^n$ has a nonzero coefficient in $p$. When $p = 0$, this returns the supremum of the empty set, $\perp$, which is the same as `none`. `nat_degree` is then defined to be `(degree p).get_or_else 0` : if `degree p` is $\perp$, it returns 0, and `(degree p)` otherwise.

We first show that it suffices to prove that `degree t = some m`. This follows easily from the fact that `0 < some m` implies `0 < m` (`with_bot.some_lt_some`). The proof is then by induction on `degree t`. If `degree t = none`, then a contradiction is derived, since we then have that `some m = none`, which then implies `m < m`, which is false. In the other case, we have that `degree t = some val` for some value `val`. Then by the definition of `option.get_or_else`, we get that `m = val`, and the proof follows simply from Lemma 1.

## 1.3 Some definitions

One of the fundamental concepts used in this proof is that single variable polynomials can also be thought of as multi-variable polynomials. In this section, we give the mechanism to translate between the two, as well as define the polynomials $V_w$, $V_s$, $B_w$, $V$, $H$ etc, sometimes separately as both single and multivariable polynomials.

Let us first understand the conversion between single and multivariable polynomials. The author defines `vars` to be an inductive type used to index 3-variable polynomials (we shall assume the variables are $X$, $Y$ and $Z$ throughout). They then define `singlify` to convert 3-variable polynomials to a single variable one : `singlify` replaces the coefficients $Y$ and $Z$ with 1 and leaves $X$ as it is.

On the other side, `X_poly`, `Y_poly` and `Z_poly` are $X$, $Y$ and $Z$ thought of as elements of $F[X, Y, Z]$.

We now give the definitions of various single and multivariable polynomials :

- `V_wit_sv` : Given $a_w = (a_l, \cdots, a_{n-1})$, returns $V_w(X) := \sum_{i=l}^{n-1} a_w(i)u_i(X)$ as an element of $F[X]$.
- `V_stmt_sv` : Given $a_s = (a_0, \cdots, a_{l-1})$, returns $V_s(X) := \sum_{i=0}^{l-1} a_s(i)u_i(X)$ as an element of $F[X]$.
- `V_stmt_mv` : Given $a_s = (a_0, \cdots, a_{l-1})$, returns $V_s(X, Y, Z) := V_s(X)$ as an element of $F[X, Y, Z]$.
- `t_mv` : Returns $t(X, Y, Z) := t(X)$ as an element of $F[X, Y, Z]$.
- `crs_powers_of_t` : Given $i \in \{0, \cdots, m-1\}$, returns $X^i$ as an element of $F[X, Y, Z]$.
- `crs_g` : Returns $Z$ as an element of $F[X, Y, Z]$.
- `crs_gb` : Returns $ZY$ as an element of $F[X, Y, Z]$.
- `crs_b_ssps` : Given $i \in \{l, \cdots, n-1\}$, returns $Yu_i(X)$ as an element of $F[X, Y, Z]$.

We also have the variables `b`, `v` and `h` which are functions/strings of length $m$, $\mathbb{Z}/m\mathbb{Z} \to F$ representing $(b_i)_{i=0}^{m-1}$, $(v_i)_{i=0}^{m-1}$ and $(h_i)_{i=0}^{m-1}$ respectively; `b'`, `v'` and `h'` which are functions/strings of length $n - l$, $\mathbb{Z}/(n-l)\mathbb{Z} \to F$ representing $(b'_i)_{i=l}^{n-l-1}$, $(v'_i)_{i=l}^{n-l-1}$ and $(h'_i)_{i=l}^{n-l-1}$ respectively; and `b_g v_g h_g b_gb v_gb h_gb`, which are elements of $F$, representing $b_\gamma, v_\gamma, h_\gamma, b_{\gamma\beta}, v_{\gamma\beta}, h_{\gamma\beta}$ respectively.

We can now define the main polynomials used :

- `B_wit` : Returns $B_w := \sum_{i=0}^{m-1} b_i X^i + b_\gamma Z + b_{\gamma\beta} YZ + \sum_{i=l}^{n-1} b'_i Y u_i(X)$ as an element of $F[X, Y, Z]$
- `V_wit` : Returns $V_w := \sum_{i=0}^{m-1} v_i X^i + v_\gamma Z + v_{\gamma\beta} YZ + \sum_{i=l}^{n-1} v'_i Y u_i(X)$ as an element of $F[X, Y, Z]$
- `H` : Returns $H := \sum_{i=0}^{m-1} h_i X^i + h_\gamma Z + h_{\gamma\beta} YZ + \sum_{i=l}^{n-1} h'_i Y u_i(X)$ as an element of $F[X, Y, Z]$
- `V` : Given $a_s = (a_0, \cdots, a_{l-1})$, returns $V := V_w + V_s$ as an element of $F[X, Y, Z]$

The above information is encapsulated in the following table :

| **Lean** | **Text** | **Description** | **Type** |
|---|---|---|---|
| `X_poly` | $X$ | $X$ | $F[X, Y, Z]$ |
| `Y_poly` | $Y$ | $Y$ | $F[X, Y, Z]$ |
| `Z_poly` | $Z$ | $Z$ | $F[X, Y, Z]$ |
| `V_wit_sv` | $V_w(X)$ | $\sum_{i=l}^{n-1} a_w(i) u_i(X)$ | $F[X]$ |
| `V_stmt_sv` | $V_s(X)$ | $\sum_{i=0}^{l-1} a_s(i) u_i(X)$ | $F[X]$ |
| `V_stmt_mv` | $V_s(X)$ | $\sum_{i=0}^{l-1} a_s(i) u_i(X)$ | $F[X, Y, Z]$ |
| `t_mv` | $t(X)$ | $t(X)$ | $F[X, Y, Z]$ |
| `crs_powers_of_t i` | $X^i$ | $X^i$ | $F[X, Y, Z]$ |
| `crs_g` | $Z$ | $Z$ | $F[X, Y, Z]$ |
| `crs_gb` | $ZY$ | $ZY$ | $F[X, Y, Z]$ |
| `crs_b_ssps i` | $Y u_i(X)$ | $F[X, Y, Z]$ | |
| `b` | $(b_i)_{i=0}^{m-1}$ | $(b_i)_{i=0}^{m-1}$ | $\mathbb{Z}/m\mathbb{Z} \to F$ |
| `v` | $(v_i)_{i=0}^{m-1}$ | $(v_i)_{i=0}^{m-1}$ | $\mathbb{Z}/m\mathbb{Z} \to F$ |
| `h` | $(h_i)_{i=0}^{m-1}$ | $(h_i)_{i=0}^{m-1}$ | $\mathbb{Z}/m\mathbb{Z} \to F$ |
| `b'` | $(b'_i)_{i=l}^{n-l-1}$ | $(b'_i)_{i=l}^{n-l-1}$ | $\mathbb{Z}/(n-l)\mathbb{Z} \to F$ |
| `v'` | $(v'_i)_{i=l}^{n-l-1}$ | $(v'_i)_{i=l}^{n-l-1}$ | $\mathbb{Z}/(n-l)\mathbb{Z} \to F$ |
| `h'` | $(h'_i)_{i=l}^{n-l-1}$ | $(h'_i)_{i=l}^{n-l-1}$ | $\mathbb{Z}/(n-l)\mathbb{Z} \to F$ |
| `b_g` | $b_\gamma$ | $b_\gamma$ | $F$ |
| `v_g` | $v_\gamma$ | $v_\gamma$ | $F$ |
| `h_g` | $h_\gamma$ | $h_\gamma$ | $F$ |
| `b_gb` | $b_{\gamma\beta}$ | $b_{\gamma\beta}$ | $F$ |
| `v_gb` | $v_{\gamma\beta}$ | $v_{\gamma\beta}$ | $F$ |
| `h_gb` | $h_{\gamma\beta}$ | $F$ | |
| `B_wit` | $B_w$ | $\sum_{i=0}^{m-1} b_i X^i + b_\gamma Z + b_{\gamma\beta} YZ + \sum_{i=l}^{n-1} b'_i Y u_i(X)$ | $F[X, Y, Z]$ |
| `V_wit` | $V_w$ | $\sum_{i=0}^{m-1} v_i X^i + v_\gamma Z + v_{\gamma\beta} YZ + \sum_{i=l}^{n-1} v'_i Y u_i(X)$ | $F[X, Y, Z]$ |
| `H` | $H$ | $\sum_{i=0}^{m-1} h_i X^i + h_\gamma Z + h_{\gamma\beta} YZ + \sum_{i=l}^{n-1} h'_i Y u_i(X)$ | $F[X, Y, Z]$ |
| `V` | $V$ | $V_s + V_w$ | $F[X, Y, Z]$ |

Finally, we say that the pair $(a_i)_{i=0}^{l-1}$ and $(a_i)_{i=l}^{n-1}$ is `satisfying` if

$$\sum_{i=0}^{l-1} a_i u_i(X) + \sum_{i=l}^{n-1} a_i u_i(X) \equiv 1 \mathtt{mod}\ t$$

that is, on dividing the above polynomial by $t$, the remainder obtained is 1. The significance of looking at these sums separately is that the witness information is only available to the prover, not the verifier.

## 1.4 Supporting lemmas

In this section we state some lemmas that shall assist us in the proof of the final theorem.

# References


[1] Bolton Bailey. Knowledge soundness of baby snarks. https://github.com/BoltonBailey/formal-snarks-project/blob/master/src/snarks/babysnark/knowledge_soundness.lean, 2021.

[2] Ye Zhang Andrew Miller and Sanket Kanjalkar. Baby snark (do do dodo dodo). https://github.com/initc3/babySNARK/blob/master/babysnark.pdf, 2020.

[3] Lean 3. https://github.com/leanprover-community/mathlib/blob/master/src/algebra/big_operators/basic.lean.