Baby SNARKs

Ashvni Narayanan, for Yatima Inc

July 19, 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 [?]. The mathematics is given in [?], 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 [?].
- 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 [?]. 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_γ	b_γ	F
`v_g`	v_γ	v_γ	F
`h_g`	h_γ	h_γ	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

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

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.

The following lemma `eq_helper` is used in `h2_1` :

Lemma 4. *Given natural numbers x and n, $x = j \iff x = j \vee (x = 0 \wedge j = 0)$*

This lemma seems obvious, however, it is quite useful to state beforehand, so it can be used directly in the next lemma. The proof is simple, we split the goal into two statements and get two goals : $x = j \to x = j \vee (x = 0 \wedge j = 0)$ and $x = j \vee (x = 0 \wedge j = 0) \to x = j$. The first implication is trivial. We must split the second implication into 2 cases : $x = j \to x = j$ and $x = 0 \wedge j = 0 \to x = j$. Both implications are trivial.

The next lemma, `h2_1` states that :

Lemma 5. *$\forall 0 \leq i < m$, the coefficient of X^i in B_w (or `B_wit`) is b_i.*

The lemma follows by tracking quotients, unfolding various definitions, removing coercions and applying the lemmas `finsupp.single_eq_single_iff`, `eq_helper` and `fin.eq_iff_veq`. This is done by applying the tactics `simp` and `unfold_coes`. For a full list of lemmas that `simp` uses, one can apply `squeeze_simp`.

Following a similar proof as above, the lemma `h3_1` is proved :

Lemma 6. *The coefficient of Z in B_w (or `B_wit`) is b_γ.*

In fact, a single `simp` proves this, with an addition of `finsupp.single_eq_single_iff`.

The lemma `h4_1` says :

Lemma 7. *Suppose that, $\forall 0 \leq i < m, b_i = 0$. Then, $b_i X^i = 0$. Equivalently, the function defined as $f(i) := b_i \cdot X^i$ is the same as the zero function.*

The lemma is stated in the function form. Here, $\cdot$ represents scalar multiplication of F on $F[X]$. The proof uses the tactic `ext`, which says that functions f and g are equal if and only if $\forall x, f(x) = g(x)$. The conclusion follows from using the hypothesis and applying `zero_smul`.

The lemma `h5_1` says :

Lemma 8. $b_{\gamma\beta} \cdot ZY = Y(b_{\gamma\beta} \cdot Z)$

The lemma uses the fact `mv_polynomial.smul_eq_C_mul`, which says that scalar multiplication of a polynomial by a constant in F is the same as multiplication of the polynomial by the constant polynomial, that is $b \cdot p(X) = b(X) * p(X)$, where $b \in F$ and a polynomial $p(X) \in F[X]$. The tactic `ring` then finishes the proof by using associativity and commutativity of multiplication. One can check what `ring` does by looking at `show_term{ring}`.

The lemma `h6_2` says :

Lemma 9. *The coefficient of Z^2 in $Ht + 1$ is 0.*

The coefficient of Z^2 in $Ht+1$ is precisely the coefficient of Z^2 in Ht, which is the same as $\sum_{i=0}^{2} coeff_H(Z^i) coeff_t(Z^{2-i})$. We know that $coeff_t(Z^i)$ is 0 for every i, which concludes the proof.

The lemma `h6_3` says :

Lemma 10. *Given $(a_i)_{i=0}^{l-1}$, the coefficient of Z^2 in $(b_{\gamma\beta} \cdot Z + \sum_{i=0}^{l-1} a_i u_i(X) + \sum_{j=l}^{n-1} b'_i u_i(X))^2$ is $b^2_{\gamma\beta}$.*

Mathematically, this statement is trivial, simply by looking at the coefficients. The code relies on first computing the power. This is done by looking at $z^2 = z * z$. One then uses `mv_polynomial.coeff_mul` to write out the coefficients in terms of sums over the `antidiagonal`, which is the same as using the binomial theorem. Given a given a finitely supported function s taking values in the natural numbers, `antidiagonal s` is the set $\{(m, n) | m + n = s\}$.