Noname manuscript No.
(will be inserted by the editor)

Evaluating Winding Numbers and Counting Complex Roots through Cauchy Indices in Isabelle/HOL

Wenda Li · Lawrence C. Paulson

the date of receipt and acceptance should be inserted later

Abstract In complex analysis, the winding number measures the number of times a path (counter-clockwise) winds around a point, while the Cauchy index can approximate how the path winds. We formalise this approximation in the Isabelle theorem prover, and provide a tactic to evaluate winding numbers through Cauchy indices. By further combining this approximation with the argument principle, we are able to make use of remainder sequences to effectively count the number of complex roots of a polynomial within some domains, such as a rectangular box and a half-plane.

Keywords Interactive theorem proving · Isabelle/HOL · computer algebra · Cauchy index · winding number · root counting · the Routh-Hurwitz stability criterion

1 Introduction

The winding number, given by

$$n(\gamma, z) = \frac{1}{2\pi i} \oint_{\gamma} \frac{dw}{w - z},$$

measures how the path γ winds around the complex point z. It is an important object in complex analysis, and its evaluation is ubiquitous among analytic proofs.

However, when formally evaluating the winding number in proof assistants such as Isabelle/HOL and HOL Light, unexpected difficulties arise, as pointed out

The first author was funded by the China Scholarship Council, via the CSC Cambridge Scholarship programme. This development is also supported by the European Research Council Advanced Grant ALEXANDRIA (Project 742178).

Wenda Li
Computer Laboratory, University of Cambridge
E-mail: wl302@cam.ac.uk

Lawrence C. Paulson
Computer Laboratory, University of Cambridge
E-mail: lp15@cam.ac.uk

by Harrison [8] and Li et al. [14]. To address this problem, we formalise a theory of the Cauchy index on the complex plane, thereby approximating how the path winds. When the path is a cycle and comprises line segments and parts of circles, we can now evaluate the winding number by calculating Cauchy indices along those sub-paths.

In addition, by further combining our previous formalisation of the argument principle [14] (which associates the winding number with the number of complex roots), we build effective procedures to count the complex roots of a polynomial within some domains, such as a rectangle box and a half-plane.

In short, the main contributions of this paper are

- a novel tactic to enable users to evaluate the winding number through Cauchy indices,
- and novel verified procedures to count complex roots of a polynomial.

The Isabelle sources of this paper are available from the Archive of Formal Proofs [11,12].

Formulations in this paper, such as the definition of the Cauchy index and statements of some key lemmas, mainly follow Rahman and Schmeisser's book [19, Chapter 11] and Eisermann's paper [6]. Nevertheless, we were still obliged to devise some proofs on our own, as discussed later.

This paper continues as follows: we start with a motivating example (§2) to explain the difficulty of formal evaluation of the winding number in Isabelle/HOL. We then present an intuitive description of the link between the winding number and the Cauchy indices (§3), which is then developed formally (§4). Next, we present verified procedures that count the number of complex roots in a domain (§5), along with some limitations (§6) and make some general remarks on the formalisation (§7). Finally, we discuss related work (§8) and present conclusions (§9).

2 A Motivating Example

In the formalisation of Cauchy's residue theorem [14], we demonstrated an application of this theorem to formally evaluate an improper integral in Isabelle/HOL:

$$\int_{-\infty}^{\infty} \frac{dx}{x^2+1} = \pi. \tag{1}$$

The idea is to embed this integral into the complex plane, and, as illustrated in Fig. 1, to construct a linear path L_r from $-r$ to r and a semi-circular path C_r centred at 0 with radius $r > 1$:

$$C_r(t) = re^{i\pi t} \qquad \text{for} \quad t \in [0,1],$$

$$L_r(t) = (1-t)(-r) + tr \qquad \text{for} \quad t \in [0,1].$$

Next, by letting

$$f(w) = \frac{1}{w^2+1},$$

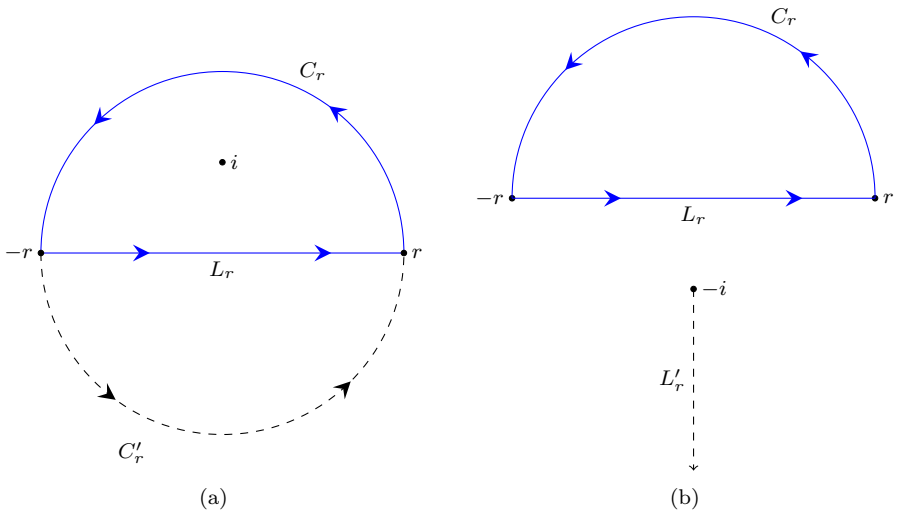

Fig. 1: Complex points $(0, -i)$ and $(0, i)$, and a closed path $L_r + C_r$

and $r \to \infty$, we can derive (1) through the following steps:

$$\int_{-\infty}^{\infty} \frac{dx}{x^2+1} = \oint_{L_r} f \tag{2}$$

$$= \oint_{L_r+C_r} f \tag{3}$$

$$= n(L_r + C_r, i)\mathrm{Res}(f, i) + n(L_r + C_r, -i)\mathrm{Res}(f, -i) \tag{4}$$

$$= \pi. \tag{5}$$

Here $L_r + C_r$ is formed by appending C_r to the end of L_r, and $\mathrm{Res}(f, i)$ is the residue of f at i. Equation (3) is because $\oint_{C_r} f = 0$ as $r \to \infty$. The application of the residue theorem is within (4); we exploit the fact that i and $-i$ are the only two singularities of f over the complex plane, since

$$\frac{1}{w^2+1} = \frac{1}{(w-i)(w+i)}.$$

While carrying out the formal proofs of (5), surprisingly, the most troublesome part of the proof is to evaluate the winding numbers:

$$n(L_r + C_r, i) = 1 \tag{6}$$

$$n(L_r + C_r, -i) = 0. \tag{7}$$

Equations (6) and (7) are straightforward to humans, as it can be seen from Fig. 1 that $L_r + C_r$ passes counterclockwise around the point i exactly one time, and around $-i$ zero times. However, formally deriving these facts was non-trivial.

Example 1 (Proof of $n(L_r+C_r,i)=1$) We defined an auxiliary semi-circular path C'_r where

$$C'_r(t) = re^{i\pi(t+1)} \qquad \text{for} \quad t \in [0,1]$$

as can be seen in Fig. 1a. As $C_r+C'_r$ forms a (full) circular path with i lying inside the circle, we had

$$n(C_r + C'_r, i) = 1. \tag{8}$$

In addition, we further proved that $C_r + C'_r$ and $L_r + C_r$ are homotopic on the space of the complex plane except for the point i (i.e., on $\mathbb{C} - \{i\}$), and hence

$$n(L_r + C_r, i) = n(C_r + C'_r, i) \tag{9}$$

by using the following Isabelle lemma:

Lemma 1 (*winding_number_homotopic_paths*)

```
fixes z::complex and γ1 γ2::"real ⇒ complex"
assumes "homotopic_paths (-{z}) γ1 γ2"
shows "winding_number γ1 z = winding_number γ2 z"
```

where *winding_number* γ_1 *z* encodes the winding number of γ_1 around z: $n(\gamma_1, z)$, and *homotopic_paths* encodes the homotopic proposition between two paths. Putting (8) and (9) together yields $n(L_r + C_r, i) = 1$, which concludes the whole proof.

Example 2 (Proof of $n(L_r+C_r,-i)=0$) We started by defining a ray L'_r starting from $-i$ and pointing towards the negative infinity of the imaginary axis:

$$L'_r(t) = (-i) - ti \qquad \text{for} \quad t \in [0,\infty)$$

as illustrated in Fig. 1b. Subsequently, we showed that

$$L'_R \text{ does not intersect with } L_r + C_r, \tag{10}$$

and then applied the following lemma in Isabelle

Lemma 2 (*winding_number_less_1*)

```
fixes z w::complex and γ::"real ⇒ complex"
assumes "valid_path γ" and "z ∉ path_image γ" and "w ≠ z"
  and not_intersection:"⋀a::real. 0 < a ⟹ z + a*(w - z) ∉ path_image γ"
shows "|Re(winding_number γ z)| < 1"
```

where

- *valid_path* γ assumes that γ is piecewise continuously differentiable on $[0,1]$,
- *z* $\notin$ *path_image* γ asserts that z is not on the path γ,
- the assumption *not_intersection* asserts that the ray starting at $z \in \mathbb{C}$ and through $w \in \mathbb{C}$ ($\{z + a(w - z) \mid a > 0\}$) does not intersect with γ—for all $a > 0$, $z + a(w - z)$ does not lie on γ.

Note that the real part of a winding number $\mathrm{Re}(n(\gamma, z))$ measures the degree of the winding: in case of γ winding around z counterclockwise for exactly one turn, we have $n(\gamma, z) = \mathrm{Re}(n(\gamma, z)) = 1$. Essentially, Lemma 2 claims that a path γ can only wind around z for less than one turn, $|\mathrm{Re}(n(\gamma, z))| < 1$, if there is a ray starting at z and not intersecting with γ. Joining Lemma 2 with (10) leads to

$$|\mathrm{Re}(n(L_r + C_r, -i))| < 1. \tag{11}$$

Moreover, as $L_r + C_r$ is a closed path,

$$n(L_r + C_r, -i) \in \mathbb{Z} \tag{12}$$

By combining (11) and (12) , we managed to derive $n(L_r + C_r, -i) = 0$.

As can be observed in Examples 1 and 2, our proofs of $n(L_r + C_r, i) = 1$ and $n(L_r + C_r, -i) = 0$ were ad hoc, and involved the manual construction of auxiliary paths or rays (e.g. C'_R and L'_R). Similar difficulties have also been mentioned by John Harrison when formalising the prime number theorem [8]. In the next section, we will introduce an idea to systematically evaluate winding numbers.

3 The Intuition

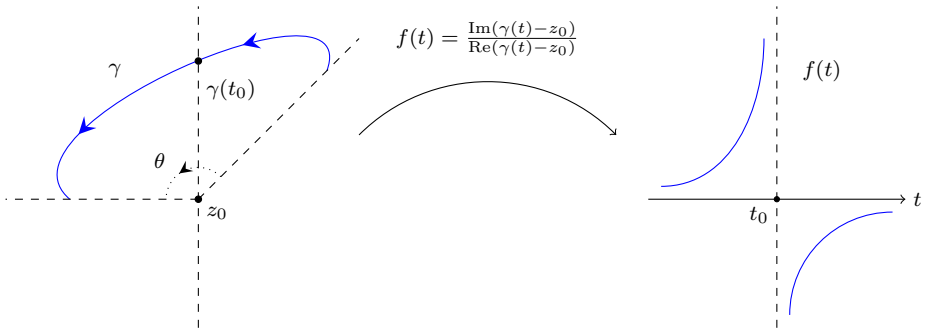

Fig. 2: Left: a path γ crosses the line $\{z \mid \mathrm{Re}(z) = \mathrm{Re}(z_0)\}$ at $\gamma(t_0)$ such that $\mathrm{Re}(\gamma(t_0)) > \mathrm{Re}(z_0)$. Right: the image of f as a point travels through γ

The fundamental idea of evaluating a winding number $n(\gamma, z_0)$ in this paper is to reduce the evaluation to *classifications* of *how* paths cross the line $\{z \mid \mathrm{Re}(z) = \mathrm{Re}(z_0)\}$: continuously or not and in which direction.

In a simple case, suppose a path γ crosses the line $\{z \mid \mathrm{Re}(z) = \mathrm{Re}(z_0)\}$ exactly once at the point $\gamma(t_0)$ such that $\mathrm{Im}(\gamma(t_0)) > \mathrm{Im}(z_0)$ (see Fig. 2 (left)), and let θ be the change in the argument of a complex point travelling through γ. It should not be hard to observe that

$$0 < \theta < 2\pi,$$

and by considering $\mathrm{Re}(n(\gamma, z_0)) = \theta/(2\pi)$ we can have

$$0 < \mathrm{Re}(n(\gamma, z_0)) < 1,$$

which is an approximation of $\mathrm{Re}(n(\gamma, z_0))$. That is, we have approximated $\mathrm{Re}(n(\gamma, z_0))$ by the way that γ crosses the line $\{z \mid \mathrm{Re}(z) = \mathrm{Re}(z_0)\}$.

To make this idea more precise, let

$$f(t) = \frac{\mathrm{Im}(\gamma(t) - z_0)}{\mathrm{Re}(\gamma(t) - z_0)}.$$

The image of f as a point travels through γ is as illustrated in Fig. 2 (right), where f jumps from $+\infty$ to $-\infty$ across t_0. We can then formally characterise those jumps.

Definition 1 (Jump) For $f : \mathbb{R} \to \mathbb{R}$ and $x \in \mathbb{R}$, we define

$$\text{jump}_+(f,x) = \begin{cases} \frac{1}{2} & \text{if } \lim_{u\to x^+} f(u) = +\infty, \\ -\frac{1}{2} & \text{if } \lim_{u\to x^+} f(u) = -\infty, \\ 0 & \text{otherwise,} \end{cases}$$

$$\text{jump}_-(f,x) = \begin{cases} \frac{1}{2} & \text{if } \lim_{u\to x^-} f(u) = +\infty, \\ -\frac{1}{2} & \text{if } \lim_{u\to x^-} f(u) = -\infty, \\ 0 & \text{otherwise.} \end{cases}$$

Specifically, we can conjecture that $\text{jump}_+(f,t_0) - \text{jump}_-(f,t_0)$ captures the way that γ crosses the line $\{z \mid \text{Re}(z) = \text{Re}(z_0)\}$ in Fig. 2, hence $\text{Re}(n(\gamma,z_0))$ can be approximated using jump_+ and jump_-:

$$\left|\text{Re}(n(\gamma,z_0)) + \frac{\text{jump}_+(f,t_0) - \text{jump}_-(f,t_0)}{2}\right| < \frac{1}{2}.$$

In more general cases, we can define Cauchy indices by summing up these jumps over an interval and along a path.

Definition 2 (Cauchy index) For $f : \mathbb{R} \to \mathbb{R}$ and $a, b \in \mathbb{R}$, the Cauchy index of f over a closed interval $[a,b]$ is defined as

$$\text{Ind}_a^b(f) = \sum_{x\in[a,b)} \text{jump}_+(f,x) - \sum_{x\in(a,b]} \text{jump}_-(f,x).$$

Definition 3 (Cauchy index along a path) Given a path $\gamma : [0,1] \to \mathbb{C}$ and a point $z_0 \in \mathbb{C}$, the Cauchy index along γ about z_0 is defined as

$$\text{Indp}(\gamma,z_0) = \text{Ind}_0^1(f)$$

where

$$f(t) = \frac{\text{Im}(\gamma(t) - z_0)}{\text{Re}(\gamma(t) - z_0)}.$$

In particular, it can be checked that the Cauchy index $\text{Indp}(\gamma,z_0)$ captures the way that γ crosses the line $\{z \mid \text{Re}(z) = \text{Re}(z_0)\}$, hence leads to an approximation of $\text{Re}(n(\gamma,z_0))$:

$$\left|\text{Re}(n(\gamma,z_0)) + \frac{\text{Indp}(\gamma,z_0)}{2}\right| < \frac{1}{2}.$$

More interestingly, by further knowing that γ is a loop we can derive $\text{Re}(n(\gamma,z_0)) = n(\gamma,z_0) \in \mathbb{Z}$ and $\text{Indp}(\gamma,z_0)/2 \in \mathbb{Z}$, following which we come to the core proposition of this paper:

Proposition 1 *Given a valid path $\gamma : [0,1] \to \mathbb{C}$ and a point $z_0 \in \mathbb{C}$, such that γ is a loop and z_0 is not on the image of γ, we have*

$$n(\gamma, z_0) = -\frac{\mathrm{Indp}(\gamma, z_0)}{2}.$$

That is, under some assumptions, we can evaluate a winding number through Cauchy indices!

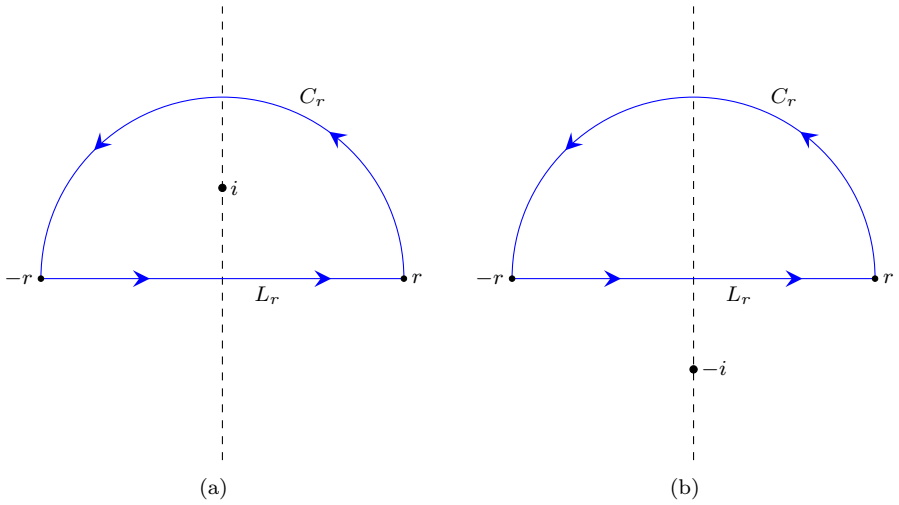

Fig. 3: Evaluating $n(L_r + C_r, i)$ and $n(L_r + C_r, -i)$ through the way that the path $L_r + C_r$ crosses the imaginary axis

A formal proof of Proposition 1 will be introduced in §4.1. Here, given the statement of the proposition, we can have alternative proofs for $n(L_r + C_r, i) = 1$ and $n(L_r + C_r, -i) = 0$.

Example 3 (Alternative proof of $n(L_r + C_r, i) = 1$) As $L_r + C_r$ is a loop, applying Proposition 1 yields

$$n(L_r + C_r, i) = -\frac{\mathrm{Indp}(L_r + C_r, i)}{2} = -\frac{1}{2}(\mathrm{Indp}(L_r, i) + \mathrm{Indp}(C_r, i)),$$

which reduces $n(L_r + C_r, i)$ to the evaluations of $\mathrm{Indp}(L_r, i)$ and $\mathrm{Indp}(C_r, i)$. In this case, by definition we can easily decide $\mathrm{Indp}(L_r, i) = -1$ and $\mathrm{Indp}(C_r, i) = -1$ as illustrated in Fig. 3a. Hence, we have

$$n(L_r + C_r, i) = -\frac{1}{2}((-1) + (-1)) = 1$$

and conclude the proof.

Example 4 (Alternative proof of $n(L_r + C_r, -i) = 0$) As shown in Fig. 3b, we can similarly have

$$\begin{aligned} n(L_R + C_R, -i) &= -\frac{\text{Indp}(L_r + C_r, -i)}{2} \\ &= -\frac{1}{2}(\text{Indp}(L_r, -i) + \text{Indp}(C_r, -i)) \\ &= -\frac{1}{2}(1 + (-1)) = 0 \end{aligned}$$

by which the proof is completed.

Compared to the previous proofs presented in Examples 1 and 2, the alternative proofs in Examples 3 and 4 are systematic and less demanding to devise once we have a formalisation of Proposition 1, which is what we will introduce in the next section.

4 Evaluating Winding Numbers

The previous section presented an informal intuition to systematically evaluate winding numbers; in this section, we will report the formal development of this intuition. We will first present a mechanised proof of Proposition 1 (§4.1), which includes mechanised definitions of jumps and Cauchy indices (i.e., Definition 1, 2 and 3) and several related properties of these objects. After that, we build a tactic in Isabelle/HOL that is used to mechanise proofs presented in Example 3 and 4 (§4.2). Finally, we discuss some subtleties we encountered during the formalisation (§4.3).

4.1 A Formal Proof of Proposition 1

For jump_- and jump_+ (see Definition 1), we have used the filter mechanism [9] to define a function `jumpF`:

```
definition jumpF::"(real ⇒ real) ⇒ real filter ⇒ real" where
  "jumpF f F ≡ (if (LIM x F. f x :> at_top) then 1/2 else
                if (LIM x F. f x :> at_bot) then -1/2 else 0)"
```

and encoded $\text{jump}_-(f, x)$ and $\text{jump}_+(f, x)$ as

`jumpF f (at_left x)` and `jumpF f (at_right x)`,

respectively. Here, `at_left x`, `at_right x`, `at_top`, and `at_bot` are all filters, where a filter is a predicate on predicates that satisfies certain properties. Filters are extensively used in the analysis library of Isabelle to encode varieties of logical quantification: for example, `at_left x` encodes the statement "for a variable that is sufficiently close to x from the left", and `at_top` represents "for a sufficiently large variable". Furthermore, `LIM x (at_left x). f x :> at_top` encoded the proposition

$$\lim_{u \to x^-} f(u) = +\infty, \tag{13}$$

and this encoding can be justified by the following equality in Isabelle:

```
(LIM x (at_left x). f x :> at_top) = (∀z. ∃b<x. ∀y>b. y < x ⟶ z ≤ f y)
```

where $\forall z.\ \exists b{<}x.\ \forall y{>}b.\ y < x \longrightarrow z \leq f\ y$ matches the usual definition of (13) in textbooks.

We can then encode $\mathrm{Ind}_a^b(f)$ and $\mathrm{Indp}(\gamma, z_0)$ (see Definitions 2 and 3) as *cindexE* and *cindex_pathE* respectively:

```
definition  cindexE::"real ⇒ real ⇒ (real ⇒ real) ⇒ real" where
  "cindexE a b f =
      (∑x∈{x. jumpF f (at_right x) ≠ 0 ∧ a ≤ x ∧ x < b}. jumpF f (at_right x))
      - (∑x∈{x. jumpF f (at_left x) ≠ 0 ∧ a < x ∧ x ≤ b}. jumpF f (at_left x))"

definition  cindex_pathE::"(real ⇒ complex) ⇒ complex ⇒ real" where
  "cindex_pathE γ z₀ = cindexE 0 1 (λt. Im (γ t - z₀) / Re (γ t - z₀))"
```

Note, in the definition of $\mathrm{Ind}_a^b(f)$ we have a term

$$\sum_{x\in[a,b)} \mathrm{jump}_+(f,x)$$

which actually hides an assumption: that only a finite number of points within the interval $[a,b)$ contribute to the sum. This assumption is made explicit when *cindexE* is defined by summing jumps over the following set:

```
{x. jumpF f (at_right x) ≠ 0 ∧ a ≤ x ∧ x < b}.
```

If the set above is infinite (i.e., the sum $\sum_{x\in[a,b)} \mathrm{jump}_+(f,x)$ is not mathematically well-defined) we have

```
(∑x∈{x. jumpF f (at_right x) ≠ 0 ∧ a ≤ x ∧ x < b}. jumpF f (at_right x)) = 0.
```

In other words, Isabelle/HOL deems the sum over an infinite set to denote zero.

Due to the issue of well-defined sums, many of our lemmas related to *cindexE* should assume a finite number of jumps:

```
definition  finite_jumpFs::"(real ⇒ real) ⇒ real ⇒ real ⇒ bool" where
  "finite_jumpFs f a b = finite {x. (jumpF f (at_left x) ≠ 0
                          ∨ jumpF f (at_right x) ≠ 0) ∧ a ≤ x ∧ x ≤ b}"
```

which guarantees the well-definedness of *cindexE*.

Now, suppose that we know that Indp is well-defined: there are only a finite number of jumps over the path. What strategy can we employ to formally prove Proposition 1? Naturally, we may want to divide the path into a finite number of segments (subpaths) separated by those jumps, and then perform inductions on these segments. To formalise the finiteness of such segments, we defined an inductive predicate:

```
inductive  finite_Psegments::"(real ⇒ bool) ⇒ real ⇒ real ⇒ bool"
    for P where
  emptyI: "a≥b ⟹ finite_Psegments P a b"|
  insertI_1: "⟦s∈{a..<b}; s=a ∨ P s; ∀t∈{s<..<b}. P t;
      finite_Psegments P a s⟧ ⟹ finite_Psegments P a b"|
  insertI_2: "⟦s∈{a..<b}; s=a ∨ P s; ∀t∈{s<..<b}. ¬P t;
      finite_Psegments P a s⟧ ⟹ finite_Psegments P a b"

definition  finite_ReZ_segments::"(real ⇒ complex) ⇒ complex ⇒ bool" where
  "finite_ReZ_segments γ z₀ = finite_Psegments (λt. Re (γ t - z₀) = 0) 0 1"
```

The idea behind *finite_ReZ_segments* is that a jump of

$$f(t) = \frac{\mathrm{Im}(\gamma(t) - z_0)}{\mathrm{Re}(\gamma(t) - z_0)}$$

takes place only if $\lambda t.\ \mathrm{Re}(\gamma(t) - z_0)$ changes from 0 to $\neq 0$ (or vice versa). Hence, each of the segments of the path γ separated by those jumps has either $\lambda t.\ \mathrm{Re}(\gamma(t) - z_0) = 0$ or $\lambda t.\ \mathrm{Re}(\gamma(t) - z_0) \neq 0$.

As can be expected, the finiteness of jumps over a path can be derived by the finiteness of segments:

Lemma 3 (*finite_ReZ_segments_imp_jumpFs*)

```
fixes γ::"real ⇒ complex" and z₀::complex
assumes "finite_ReZ_segments γ z₀" and "path γ"
shows "finite_jumpFs (λt. Im (γ t - z₀)/Re (γ t - z₀)) 0 1"
```

where *path* γ asserts that γ is a continuous function on [0..1] (so that it is a path). Roughly speaking, Lemma 3 claims that a path will have a finite number of jumps if it can be divided into a finite number of segments.

By assuming such a finite number of segments we have well-defined *cindex_pathE*, and can then derive some useful related properties:

Lemma 4 (*cindex_pathE_subpath_combine*)

```
fixes γ::"real ⇒ complex" and z₀::complex
assumes "finite_ReZ_segments γ z₀"and "path γ"
  and "0≤a" and "a≤b" and "b≤c" and "c≤1"
shows "cindex_pathE (subpath a b γ) z₀ + cindex_pathE (subpath b c γ) z₀
       = cindex_pathE (subpath a c γ) z₀"
```

where *subpath a b* γ gives a sub-path of γ based on parameters a and b:

```
definition  subpath :: "real ⇒ real ⇒ (real ⇒ 'a) ⇒ real
   ⇒ 'a::real_normed_vector"
  where "subpath a b γ ≡ (λt. γ((b - a) * t + a))"
```

Essentially, Lemma 4 indicates that we can combine Cauchy indices along consecutive parts of a path: given a path γ and three parameters a, b, c with $0 \leq a \leq b \leq c \leq 1$, we have

$$\mathrm{Indp}(\gamma_1, z_0) + \mathrm{Indp}(\gamma_2, z_0) = \mathrm{Indp}(\gamma_3, z_0).$$

where $\gamma_1 = \lambda t.\, \gamma((b-a)t + a)$, $\gamma_2 = \lambda t.\, \gamma((c-b)t + b)$ and $\gamma_3 = \lambda t.\, \gamma((c-a)t + a)$.

More importantly, we now have an induction rule for a path with a finite number of segments:

Lemma 5 (*finite_ReZ_segments_induct*)

```
fixes γ::"real ⇒ complex" and z₀::complex
  and P::"(real ⇒ complex) ⇒ complex ⇒ bool"
assumes "finite_ReZ_segments γ z₀"
  and sub0:"⋀g z. (P (subpath 0 0 g) z)"
  and subEq:"(⋀s g z. ⟦s ∈ {0..<1}; s=0 ∨ Re (g s) = Re z;
       ∀t ∈ {s<..<1}. Re (g t) = Re z;
       finite_ReZ_segments (subpath 0 s g) z;
       P (subpath 0 s g) z⟧ ⟹ P g z)"
  and subNEq:"(⋀s g z. ⟦s ∈ {0..<1}; s=0 ∨ Re (g s) = Re z;
       ∀t ∈ {s<..<1}. Re (g t) ≠ Re z;
       finite_ReZ_segments (subpath 0 s g) z;
       P (subpath 0 s g) z⟧ ⟹ P g z)"
shows "P γ z₀"
```

where `P` is a predicate that takes a path γ and a complex point z_0, and

- `sub0` is the base case that `P` holds for a constant path;
- `subEq` is the inductive case when the last segment is right on the line $\{x \mid \mathrm{Re}(x) = \mathrm{Re}(z)\}$: $\forall t \in (s, 1).\ \mathrm{Re}(g(t)) = \mathrm{Re}(z)$;
- `subNEq` is the inductive case when the last segment does not cross the line $\{x \mid \mathrm{Re}(x) = \mathrm{Re}(z)\}$: $\forall t \in (s, 1).\ \mathrm{Re}(g(t)) \neq \mathrm{Re}(z)$.

Given a path γ with a finite number of segments, a complex point z_0 and a predicate P that takes a path and a complex number and returns a boolean, Lemma 5 provides us with an inductive rule to derive $P(\gamma, z_0)$ by recursively examining the last segment.

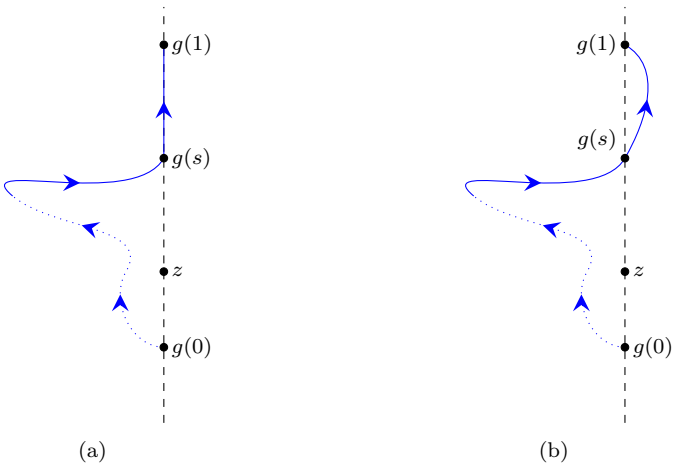

Fig. 4: Inductive cases when applying Lemma 5

Before attacking Proposition 1, we can show an auxiliary lemma about $\mathrm{Re}(n(\gamma, z_0))$ and $\mathrm{Indp}(\gamma, z_0)$ when the end points of γ are on the line $\{z \mid \mathrm{Re}(z) = \mathrm{Re}(z_0)\}$:

Lemma 6 (`winding_number_cindex_pathE_aux`)
fixes γ`::"real ⇒ complex"` **and** z_0 `:: complex`
assumes `"finite_ReZ_segments` γ z_0`"` **and** `"valid_path` γ`"`
and `"`$z_0 \notin$ `path_image` γ`"` **and** `"Re (`γ `1) = Re` z_0`"`
and `"Re (`γ `0) = Re` z_0`"`
shows `"2 * Re(winding_number` γ z_0`) = - cindex_pathE` γ z_0`"`

Here, Lemma 6 is almost equivalent to Proposition 1 except for that more restrictions haven been placed on the end points of γ.

Proof of Lemma 6. As there are a finite number of segments along γ (i.e., `finite_ReZ_segments` γ z_0), by inducting on these segments with Lemma 5 we end up with three cases. The base case is straightforward: given a constant path $g : [0, 1] \to \mathbb{C}$ and a complex point $z \in \mathbb{C}$, we have $\mathrm{Re}(n(g, z)) = 0$ and $\mathrm{Indp}(g, z) = 0$, hence $2\,\mathrm{Re}(n(g, z)) = -\,\mathrm{Indp}(g, z)$.

For the inductive case when the last segment is right on the line $\{x \mid \mathrm{Re}(x) = \mathrm{Re}(z)\}$, there is $\forall t \in (s, 1).\ \mathrm{Re}(g(t)) = \mathrm{Re}(z)$ as illustrated in Fig. 4a. Let

$$g_1(t) = g(st)$$

$$g_2(t) = g((1-s)t).$$

We have

$$n(g, z) = n(g_1, z) + n(g_2, z), \tag{14}$$

and, by the induction hypothesis,

$$2\,\mathrm{Re}(n(g_1, z)) = -\,\mathrm{Indp}(g_1, z). \tag{15}$$

Moreover, it is possible to derive

$$2\,\mathrm{Re}(n(g_2, z)) = -\,\mathrm{Indp}(g_2, z), \tag{16}$$

since $n(g_2, z) = 0$ and $\mathrm{Indp}(g_2, z) = 0$. Furthermore, by Lemma 4 we can sum up the Cauchy index along g_1 and g_2:

$$\mathrm{Indp}(g_1, z) + \mathrm{Indp}(g_2, z) = \mathrm{Indp}(g, z) \tag{17}$$

Combining Equations (14), (15), (16) and (17) yields

$$\begin{aligned} 2\,\mathrm{Re}(n(g, z)) &= 2(\mathrm{Re}(n(g_1, z)) + \mathrm{Re}(n(g_2, z))) \\ &= -\,\mathrm{Indp}(g_1, z) - \mathrm{Indp}(g_2, z) \\ &= -\,\mathrm{Indp}(g, z) \end{aligned} \tag{18}$$

which concludes the case.

For the other inductive case when the last segment does not cross the line $\{x \mid \mathrm{Re}(x) = \mathrm{Re}(z)\}$, without loss of generality, we assume

$$\forall t \in (s, 1).\ \mathrm{Re}(g(t)) > \mathrm{Re}(z), \tag{19}$$

and the shape of g is as illustrated in Fig. 4b. Similar to the previous case, by letting $g_1(t) = g(st)$ and $g_2(t) = g((1-s)t)$, we have $n(g, z) = n(g_1, z) + n(g_2, z)$ and, by the induction hypothesis, $2\,\mathrm{Re}(n(g_1, z)) = -\,\mathrm{Indp}(g_1, z)$. Moreover, by observing the shape of g_2 we have

$$2\,\mathrm{Re}(n(g_2, z)) = \mathrm{jump}_-(f, 1) - \mathrm{jump}_+(f, 0) \tag{20}$$

$$\mathrm{Indp}(g_2, z) = \mathrm{jump}_+(f, 0) - \mathrm{jump}_-(f, 1) \tag{21}$$

where $f(t) = \mathrm{Im}(g_2(t) - z)/\mathrm{Re}(g_2(t) - z)$. Combining (20) with (21) leads to $2\,\mathrm{Re}(n(g_2, z)) = -\,\mathrm{Indp}(g_2, z)$, following which we finish the case by deriving $2\,\mathrm{Re}(n(g, z)) = -\,\mathrm{Indp}(g, z)$ in a way analogous to (18). □

Finally, we are ready to formally derive Proposition 1 in Isabelle/HOL:

Theorem 1 (*winding_number_cindex_pathE*)

```
fixes γ::"real ⇒ complex" and z₀::complex
assumes "finite_ReZ_segments γ z₀" and "valid_path γ"
    and "z₀ ∉ path_image γ" and "γ 0 = γ 1"
shows "winding_number γ z₀ = - cindex_pathE γ z₀ / 2"
```

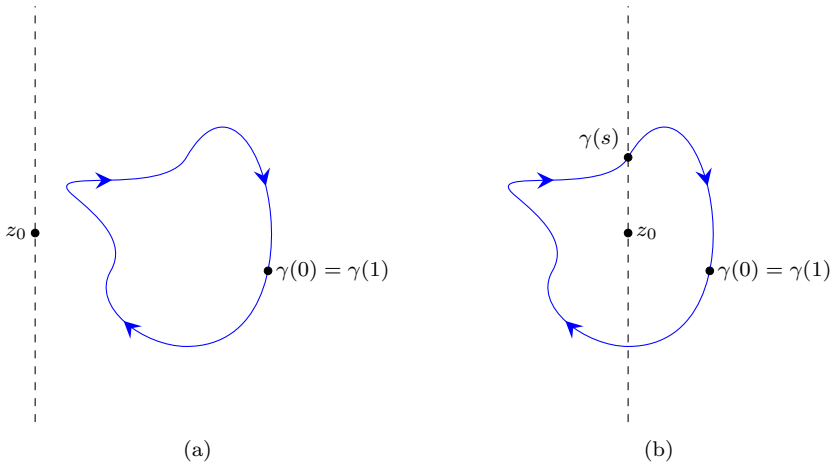

Fig. 5: To derive $n(\gamma, z_0) = -\frac{\mathrm{Indp}(\gamma, z_0)}{2}$ when γ is a loop

Proof. By assumption, we know that γ is a loop, and the point $\gamma(0) = \gamma(1)$ can be away from the line $\{z \mid \mathrm{Re}(z) = \mathrm{Re}(z_0)\}$ which makes Lemma 6 inapplicable. To resolve this problem, we look for a point $\gamma(s)$ on γ such that $0 \leq s \leq 1$ and $\mathrm{Re}(\gamma(s)) = \mathrm{Re}(z_0)$, and we can either fail or succeed.

In the case of failure, without loss of generality, we can assume $\mathrm{Re}(\gamma(t)) > \mathrm{Re}(z_0)$ for all $0 \leq t \leq 1$, and the shape of γ is as illustrated in Fig. 5a. As the path γ does not cross the line $\{z \mid \mathrm{Re}(z) = \mathrm{Re}(z_0)\}$, we can evaluate

$$\mathrm{Indp}(\gamma, z_0) = 0$$

$$n(\gamma, z_0) = \mathrm{Re}(n(\gamma, z_0)) = \frac{\mathrm{Im}(\mathrm{Ln}(\gamma(1) - z_0)) - \mathrm{Im}(\mathrm{Ln}(\gamma(0) - z_0))}{2\pi} = 0$$

where Ln is the principle value of a complex logarithm function with its branch being the negative real axis and $-\pi < \mathrm{Im}(Ln(z)) \leq \pi$ for all z. Hence, $n(\gamma, z_0) = -\mathrm{Indp}(\gamma, z_0)/2$ which concludes the case.

In the case of success, as illustrated in Fig. 5b, we have $\mathrm{Re}(\gamma(s)) = \mathrm{Re}(z_0)$. We then define a shifted path γ_s:

$$\gamma_s(t) = \begin{cases} \gamma(t+s) & \text{if } s + t \leq 1, \\ \gamma(t+s-1) & \text{otherwise,} \end{cases}$$

such that $\mathrm{Re}(\gamma_s(0)) = \mathrm{Re}(\gamma_s(1)) = \mathrm{Re}(z_0)$. By applying Lemma 6, we obtain a relationship between $\mathrm{Re}(n(\gamma_s, z_0))$ and $\mathrm{Indp}(\gamma_s, z_0)$:

$$2\,\mathrm{Re}(n(\gamma_s, z_0)) = -\,\mathrm{Indp}(\gamma_s, z_0),$$

following which we have $n(\gamma, z_0) = -\,\mathrm{Indp}(\gamma, z_0)/2$, since $n(\gamma_s, z_0) = n(\gamma, z_0)$ and $\mathrm{Indp}(\gamma_s, z_0) = \mathrm{Indp}(\gamma, z_0)$. □

4.2 A Tactic for Evaluating Winding Numbers

With Proposition 1 formalised, we are now able to build a tactic to evaluate winding numbers using Cauchy indices. The idea has already been sketched in Examples 3 and 4. We have built a tactic `eval_winding`, for goals of the form

$$n(\gamma_1 + \gamma_2 + \cdots + \gamma_n, z_0) = k, \tag{22}$$

where k is an integer and γ_j $(1 \leq j \leq n)$ is either a linear path:

$$\gamma_j(t) = (1-t)a + tb \quad \text{where } a, b \in \mathbb{C}$$

or a part of a circular path:

$$\gamma_j(t) = z + re^{i((1-t)a+tb)} \quad \text{where } a, b, r \in \mathbb{R} \text{ and } z \in \mathbb{C}.$$

The tactic `eval_winding` will transform (22) into

$$\gamma_j(1) = \gamma_{j+1}(0) \text{ for all } 1 \leq j \leq n-1, \text{ and } \gamma_n(1) = \gamma_1(0), \tag{23}$$

$$z_0 \notin \{\gamma_j(t) \mid 0 \leq t \leq 1\} \text{ for all } 1 \leq j \leq n, \tag{24}$$

$$\mathrm{Indp}(\gamma_1, z_0) + \mathrm{Indp}(\gamma_2, z_0) + \cdots + \mathrm{Indp}(\gamma_n, z_0) = -2k, \tag{25}$$

where (23) ensures that the path $\gamma_1 + \gamma_2 + \cdots + \gamma_n$ is a loop; (24) certifies that z_0 is not on the image of $\gamma_1 + \gamma_2 + \cdots + \gamma_n$.

To achieve this transformation, `eval_winding` will first perform a substitution step on the left-hand side of Equation (22) using Theorem 1. As the substitution is conditional, we will need to resolve four extra subgoals (i.e., (26), (27), (28) and (29) as follows) and Equation (22) is transformed into (30):

$$\texttt{finite\_ReZ\_segments (}\gamma_1 \texttt{ +++ } \gamma_2 \texttt{ +++ ... +++}\gamma_n\texttt{) } z_0, \tag{26}$$

$$\texttt{valid\_path (}\gamma_1 \texttt{ +++ } \gamma_2 \texttt{ +++ ... +++}\gamma_n\texttt{)}, \tag{27}$$

$$z_0 \notin \texttt{ path\_image (}\gamma_1 \texttt{ +++ } \gamma_2 \texttt{ +++ ... +++}\gamma_n\texttt{)}, \tag{28}$$

$$\texttt{(}\gamma_1 \texttt{ +++ } \gamma_2 \texttt{ +++ ... +++}\gamma_n\texttt{) 0 = (}\gamma_1 \texttt{ +++ } \gamma_2 \texttt{ +++ ... +++}\gamma_n\texttt{) 1}, \tag{29}$$

$$\texttt{- cindex\_pathE (}\gamma_1 \texttt{ +++ } \gamma_2 \texttt{ +++ ... +++}\gamma_n\texttt{) } z_0 \texttt{ / 2 = k}. \tag{30}$$

To simplify (26), the tactic will keep applying the following introduction rule:[1]

Lemma 7 (`finite_ReZ_segments_joinpaths`)

```
fixes γ1 γ2 :: "real ⇒ complex" and z0 :: complex
assumes "finite_ReZ_segments γ1 z0" and "finite_ReZ_segments γ2 z0"
  and "path γ1" and "path γ2" and "γ1 1 = γ2 0"
shows "finite_ReZ_segments (γ1+++γ2) z0"
```

to eliminate the path join operations (`+++`) until the predicate `finite_ReZ_segments` is only applied to a linear path or a part of a circular path, and either of these two cases can be directly discharged because these two kinds of paths are proved to be divisible into a finite number of segments by the imaginary axis:

[1] Applying an introduction rule will replace a goal by a set of subgoals derived from the premises of the rule, provided the goal can be unified with the conclusion of the rule.

Lemma 8 (*finite_ReZ_segments_linepath*)
```
"finite_ReZ_segments (linepath a b) z"
```

Lemma 9 (*finite_ReZ_segments_part_circlepath*)
```
"finite_ReZ_segments (part_circlepath z0 r st tt) z"
```

In terms of other subgoals introduced when applying Lemma 7, such as *path* γ_1, *path* γ_2 and γ_1 *1 =* γ_2 *0*, we can discharge them by the following introduction and simplification rules (all of which have been formally proved):

- ⟦*path* γ_1*; path* γ_2*;* γ_1 *1 =* γ_2 *0*⟧ ⟹ *path(*γ_1 *+++* γ_2*)*,
- *path (part_circlepath* z_0 *r st tt)*,
- *path (linepath a b)*,
- *(*γ_1 *+++* γ_2*) 1 =* γ_2 *1*,
- *(*γ_1 *+++* γ_2*) 0 =* γ_1 *0*.

As a result, *eval_winding* will eventually simplify the subgoal (26) to (23).

Similar to the process of simplifying (26) to (23), the tactic *eval_winding* will also simplify

- (27) to (23),
- (28) to (24),
- and (29) to (23).

Finally, with respect to (30), we can similarly rewrite with a rule between the Cauchy index (*cindex_pathE*) and the path join operation (*+++*):

Lemma 10 (*cindex_pathE_joinpaths*)
```
fixes γ1 γ2 :: "real ⇒ complex" and z0 :: complex
assumes "finite_ReZ_segments γ1 z0" and "finite_ReZ_segments γ2 z0"
  and "path γ1" and "path γ2" and "γ1 1 = γ2 0"
shows "cindex_pathE (γ1 +++ γ2) z0 = cindex_pathE γ1 z0 + cindex_pathE γ2 z0"
```

to convert the subgoal (30) to (23) and (25).

After building the tactic *eval_winding*, we are now able to convert a goal like Equation (22) to (23), (24) and (25). In most cases, discharging (23) and (24) is straightforward. To derive (25), we will need to formally evaluate each $\mathrm{Indp}(\gamma_j, z_0)$ $(1 \leq j \leq n)$ when γ_j is either a linear path or a part of a circular path.

When γ_j is a linear path, the following lemma grants us a way to evaluate $\mathrm{Indp}(\gamma_j, z_0)$ through its right-hand side:

Lemma 11 (*cindex_pathE_linepath*)
```
fixes a b z0 :: complex
assumes "z0∉path_image (linepath a b)"
shows "cindex_pathE (linepath a b) z0 = (
  let c1 = Re a - Re z0;
      c2 = Re b - Re z0;
      c3 = Im a * Re b + Re z0 * Im b + Im z0 * Re a - Im z0 * Re b
           - Im b * Re a - Re z0 * Im a;
      d1 = Im a - Im z0;
      d2 = Im b - Im z0
  in if (c1>0 ∧ c2<0) ∨ (c1<0 ∧ c2>0) then
       (if c3>0 then 1 else -1)
     else
       (if (c1=0 ⟷ c2≠0) ∧ (c1=0 ⟶d1≠0) ∧ (c2=0 ⟶ d2≠0) then
           if (c1=0 ∧ (c2 >0 ⟷ d1>0)) ∨ (c2=0 ∧ (c1 >0 ⟷ d2<0))
           then 1/2 else -1/2
        else 0))"
```

Although Lemma 11 may appear terrifying, evaluating its right-hand side is usually automatic when the number of free variables is small. For example, in a formal proof of Example 3 in Isabelle/HOL, we can have the following fragment:

```
lemma
  fixes R::real
  assumes "R>1"
  shows "winding_number (part_circlepath 0 R 0 pi +++ linepath (-R) R) i = 1"
proof  (winding_eval, simp_all)
  ...
  have "i ∉ path_image (linepath (- R) (R::complex))" by ...
  from cindex_pathE_linepath[OF this] ⟨R>1⟩
  have "cindex_pathE (linepath (-R) (R::complex)) i = -1" by  auto
  ...
qed
```

where `winding_eval` is first applied to convert the goal into (23), (24) and (25), and `simp_all` subsequently simplifies those newly generated subgoals. In the middle of the proof, we show that the complex point i is not on the image of the linear path L_r (i.e., `linepath (-R) (R::complex))` in Isabelle/HOL), following which we apply Lemma 11 to derive $\text{Indp}(L_r, i) = -1$: the evaluation process is automatic through the command `auto`, given the assumption `R>1`.

When γ_j is a part of a circular path, a similar lemma has been provided to facilitate the evaluation of $\text{Indp}(\gamma_j, z_0)$.

4.3 Subtleties

The first subtlety we have encountered during the formalisation of Proposition 1 is about the definitions of jumps and Cauchy indices, for which our first attempt followed the standard definitions in textbooks [2,16,19].

Definition 4 (Jump) For $f : \mathbb{R} \to \mathbb{R}$ and $x \in \mathbb{R}$, we define

$$\text{jump}(f, x) = \begin{cases} 1 & \text{if } \lim_{u\to x^-} f(u) = -\infty \text{ and } \lim_{u\to x^+} f(u) = +\infty, \\ -1 & \text{if } \lim_{u\to x^-} f(u) = +\infty \text{ and } \lim_{u\to x^+} f(u) = -\infty, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 5 (Cauchy index) For $f : \mathbb{R} \to \mathbb{R}$ and $a, b \in \mathbb{R}$, the Cauchy index of f over an open interval (a, b) is defined as

$$\text{Ind}_a^b(f) = \sum_{x\in(a,b)} \text{jump}(f, x).$$

The impact of the difference between the current definition of the Cauchy index (i.e., Definition 2) and the classic one (i.e., Definition 5) is small when formalising the Sturm-Tarski theorem [13,10], where f is a rational function. In this case, the path γ intersects with the line $\{z \mid \text{Re}(z) = \text{Re}(z_0)\}$ a finite number of times, and for each intersection point (see Fig. 6a and b), by letting $f(t) = \text{Im}(\gamma(t) - z_0)/\text{Re}(\gamma(t) - z_0)$, we have

$$\text{jump}(f, t) = \text{jump}_+(f, t) - \text{jump}_-(f, t),$$

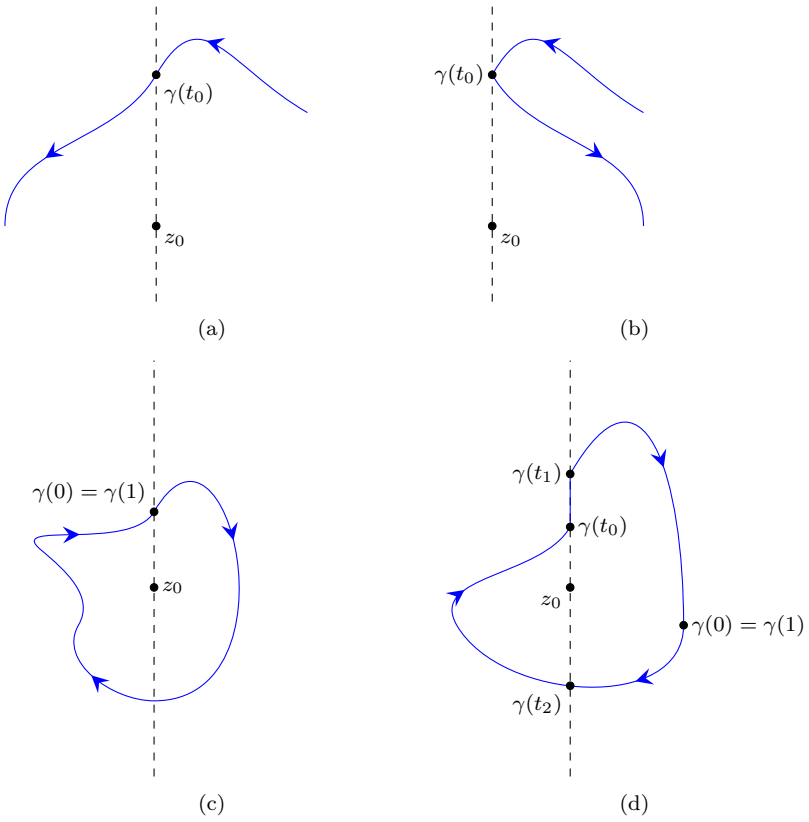

Fig. 6: Different ways a path γ can intersect with the line $\{z \mid \mathrm{Re}(z) = \mathrm{Re}(z_0)\}$

hence

$$\sum_{x \in (a,b)} \mathrm{jump}(f, x) = \sum_{x \in [a,b)} \mathrm{jump}_+(f, x) - \sum_{x \in (a,b]} \mathrm{jump}_-(f, x),$$

provided $\mathrm{jump}_+(f, a) = 0$ and $\mathrm{jump}_-(f, b) = 0$. That is, the classic Cauchy index and the current one are equal when f is a rational function and does not jump at both ends of the target interval.

Naturally, the disadvantages of Definition 5 are twofold:

- The function $\lambda t.\, \mathrm{Re}(\gamma(t) - z_0)$ cannot vanish at either end of the interval. That is, we need to additionally assume $\mathrm{Re}(\gamma(0) - z_0) \neq 0$ as in Rahman and Schmeisser's formulation [19, Lemma 11.1.1 and Theorem 11.1.3], and Proposition 1 will be inapplicable in the case of Fig. 6c where $\mathrm{Re}(\gamma(0)) = \mathrm{Re}(\gamma(1)) = \mathrm{Re}(z_0)$.
- The function $\lambda t.\, \mathrm{Im}(\gamma(t) - z_0)/\mathrm{Re}(\gamma(t) - z_0)$ has to be rational, which makes Proposition 1 inapplicable for cases like in Fig. 6d (if we follow Definition 5). To elaborate, it can be observed in Fig. 6d that $n(\gamma, z_0) = -1$, while we will only

get a wrong answer by following Definition 5 and evaluating via Proposition 1:

$$-\frac{1}{2}\left(\sum_{x\in(0,1)}\text{jump}(f,x)\right)=-\frac{\text{jump}(f,t_2)}{2}=-\frac{1}{2},$$

where $f(t)=\text{Im}(\gamma(t)-z_0)/\text{Re}(\gamma(t)-z_0)$. In comparison, Definition 2 leads to the correct answer:

$$\begin{aligned}n(\gamma,z_0)&=-\frac{1}{2}\left(\sum_{x\in[0,1)}\text{jump}_+(f,x)-\sum_{x\in(0,1]}\text{jump}_-(f,x)\right)\\&=-\frac{1}{2}\left(\text{jump}_+(f,t_2)+\text{jump}_+(f,t_1)-\text{jump}_-(f,t_2)-\text{jump}_-(f,t_0)\right)\\&=-\frac{1}{2}\left(\frac{1}{2}+\frac{1}{2}-(-\frac{1}{2})-(-\frac{1}{2})\right)\\&=-1.\end{aligned}$$

Fortunately, Michael Eisermann [6] recently proposed a new formulation of the Cauchy index that overcomes those two disadvantages, and this new formulation is what we have followed (in Definitions 1 and 2).

Another subtlety we ran into was the well-definedness of the Cauchy index. Such well-definedness is usually not an issue and left implicit in the literature, because, in most cases, the Cauchy index is only defined on rational functions, where only finitely many points can contribute to the sum. When attempting to formally derive Proposition 1, we realised that this assumption needed to be made explicit, since the path γ can be flexible enough to allow the function $f(t)=\text{Im}(\gamma(t)-z_0)/\text{Re}(\gamma(t)-z_0)$ to be non-rational (e.g. Fig. 6d). In our first attempt of following Definition 5, the Cauchy index was formally defined as follows:

```
definition cindex::"real ⇒ real ⇒ (real ⇒ real) ⇒ int" where
  "cindex a b f = (∑ x∈{x. jump f x≠0 ∧ a<x ∧ x<b}. jump f x)"
```

and its well-definedness was ensured by the finite number of times that γ crosses the line $\{z \mid \text{Re}(z)=\text{Re}(z_0)\}$:

```
definition finite_axes_cross::"(real ⇒ complex) ⇒ complex ⇒ bool" where
  "finite_axes_cross γ z₀ =
     finite {t. (Re (γ t - z₀) = 0 ∨ Im (γ t - z₀) = 0) ∧ 0 ≤ t ∧ t ≤ 1}"
```

where the part `Re (γ t - z₀) = 0` ensures that `jump f t` is non-zero only at finitely many points over the interval $[0,1]$. When constrained by `finite_axes_cross`, the function $f(t)=\text{Im}(\gamma(t)-z_0)/\text{Re}(\gamma(t)-z_0)$ behaves like a rational function. More importantly, the path γ, in this case, can be divided into a finite number of ordered segments delimited by those points over $[0,1]$, which makes an inductive proof of Proposition 1 possible. However, after abandoning our first attempt and switching to Definition 2, the well-definedness of the Cauchy index is assured by the finite number of jump_+ and jump_- of f (i.e., Definition `finite_jumpFs` in §4.1), with which we did not know how to divide the path γ into segments and carry out an inductive proof. It took us some time to properly define the assumption of a finite number of segments (i.e., Definition `finite_ReZ_segments`) that implied the well-definedness using Lemma 3 and provided a lemma for inductive proofs (i.e., Lemma 5).

5 Counting the Number of Complex Roots

The previous section described a way to evaluate winding numbers via Cauchy indices. In this section, we will further explore this idea and propose verified procedures to count the number of complex roots of a polynomial in some domain such as a rectangle and a half-plane.

Does a winding number have anything to do with the number of roots of a polynomial? The answer is yes. Thanks to the argument principle, we can calculate the number of roots by evaluating a contour integral:

$$\frac{1}{2\pi i} \oint_{\gamma} \frac{p'(x)}{p(x)} dx = N \tag{31}$$

where $p \in \mathbb{C}[x]$, $p'(x)$ is the first derivative of p and N is the number of complex roots of p (counted with multiplicity) inside the loop γ. Also, by the definition of winding numbers, we have

$$n(p \circ \gamma, 0) = \frac{1}{2\pi i} \oint_{\gamma} \frac{p'(x)}{p(x)} dx. \tag{32}$$

Combining Equations (31) and (32) gives us the relationship between a winding number and the number of roots of a polynomial:

$$n(p \circ \gamma, 0) = N. \tag{33}$$

And the question becomes: can we evaluate $n(p \circ \gamma, 0)$ via Cauchy indices?

5.1 Roots in a Rectangle

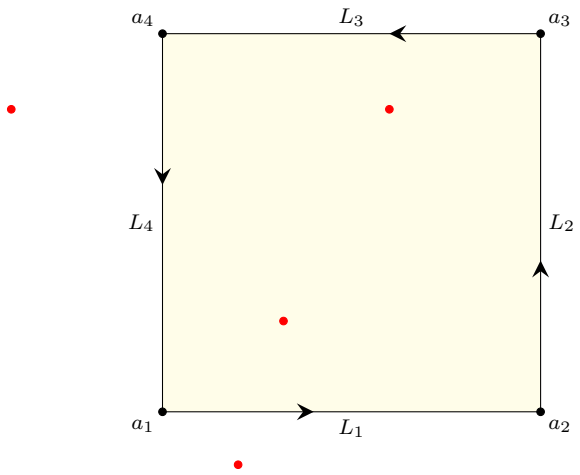

Fig. 7: Complex roots of a polynomial (red dots) and a rectangular path ($L_1 + L_2 + L_3 + L_4$) on the complex plane

Let N be the number of complex roots of a polynomial p inside the rectangle defined by its lower left corner a_1 and upper right corner a_3. As illustrated in Fig. 7, we can define four linear paths along the edge of the rectangle:

$$\begin{aligned} L_1(t) &= (1-t)a_1 + ta_2 \\ L_2(t) &= (1-t)a_2 + ta_3 \\ L_3(t) &= (1-t)a_3 + ta_4 \\ L_4(t) &= (1-t)a_4 + ta_1 \end{aligned}$$

where $a_2 = \mathrm{Re}(a_3) + i\,\mathrm{Im}(a_1)$ and $a_4 = \mathrm{Re}(a_1) + i\,\mathrm{Im}(a_3)$. Combining Proposition 1 with Equation (33) yields

$$\begin{aligned} N &= n(p \circ (L_1 + L_2 + L_3 + L_4), 0) \\ &= -\frac{1}{2}\,\mathrm{Indp}(p \circ (L_1 + L_2 + L_3 + L_4), 0) \\ &= -\frac{1}{2}\left(\mathrm{Indp}(p \circ L_1, 0) + \mathrm{Indp}(p \circ L_2, 0) + \mathrm{Indp}(p \circ L_3, 0) + \mathrm{Indp}(p \circ L_4, 0)\right). \end{aligned} \tag{34}$$

Here, the path $p \circ L_j : [0,1] \to \mathbb{C}$ $(1 \le j \le 4)$ is (mostly) neither a linear path nor a part of a circular path, which indicates that the evaluation strategies of §4.2, such as Lemma 11, will no longer apply. Thankfully, the Sturm-Tarski theorem [10,13] came to our rescue.

In general, the Sturm-Tarski theorem is about calculating Tarski queries through sign variations and signed remainder sequences: let $p, q \in \mathbb{R}[x]$, a and b be two extended real numbers such that $a < b$ and are not roots of p, we have

$$\mathrm{TaQ}(q, p, a, b) = \mathrm{Var}(\mathrm{SRemS}(p, p'q); a, b) \tag{35}$$

where

- p' is the first derivative of p,
- the Tarski query $\mathrm{TaQ}(q, p, a, b)$ defined as follows:

$$\mathrm{TaQ}(q, p, a, b) = \sum_{x \in (a,b), p(x)=0} \mathrm{sgn}(q(x)),$$

- $\mathrm{SRemS}(p, q)$ is the signed remainder sequence started with p and q.
- Let $[p_1, p_2, ..., p_n]$ be a sequence of polynomials, $\mathrm{Var}([p_1, p_2, ..., p_n]; a, b)$ is the difference in the number of sign variations when evaluating $[p_1, p_2, ..., p_n]$ at a and b:

$$\begin{aligned} &\mathrm{Var}([p_1, p_2, ..., p_n]; a, b) \\ &\qquad = \mathrm{Var}([p_1(a), p_2(a), ..., p_n(a)]) - \mathrm{Var}([p_1(b), p_2(b), ..., p_n(b)]). \end{aligned} \tag{36}$$

Note that when $q = 1$, (35) becomes the famous Sturm's theorem, which counts the number of distinct real roots over an interval. For example, by calculating

$$\begin{aligned}\mathrm{TaQ}(1, (x-1)(x-2), 0, 3) &= \mathrm{Var}(\mathrm{SRemS}(x^2 - 3x + 2, 2x - 3); 0, 3)\\ &= \mathrm{Var}([x^2 - 3x + 2, 2x - 3, 1/4]; 0, 3)\\ &= \mathrm{Var}([x^2 - 3x + 2, 2x - 3, 1/4]; 0)\\ &\quad - \mathrm{Var}([x^2 - 3x + 2, 2x - 3, 1/4]; 3)\\ &= \mathrm{Var}([2, -3, 1/4]) - \mathrm{Var}([2, 3, 1/4])\\ &= 2 - 0 = 2,\end{aligned}$$

we know that the polynomial $x^2 - 3x + 2$ has two distinct real roots within the interval $(0, 3)$.

In our previous formal proof of the Sturm-Tarski theorem [10,13], we used the Cauchy index to relate the Tarski query and the right-hand side of (35). Therefore, as a byproduct, we can also evaluate the Cauchy index through sign variations and signed remainder sequences:

$$\mathrm{Ind}_a^b\left(\lambda t.\ \frac{q(t)}{p(t)}\right) = \mathrm{Var}(\mathrm{SRemS}(p, q); a, b), \tag{37}$$

where $p, q \in \mathbb{R}[x]$, a, b are two extended real numbers such that $a < b$ and are not roots of p.

Back to the case of $\mathrm{Indp}(p \circ L_j, 0)$, we have

$$\mathrm{Indp}(p \circ L_j, 0) = \mathrm{Ind}_0^1\left(\lambda t.\ \frac{\mathrm{Im}(p(L_j(t)))}{\mathrm{Re}(p(L_j(t)))}\right),$$

and both $\mathrm{Im}(p(L_j(t)))$ and $\mathrm{Re}(p(L_j(t)))$ happen to be polynomials with real coefficients. Therefore, combining Equations (34) and (37) yields an approach to count the number of roots inside a rectangle.

While proceeding to the formal development, the first problem we encountered was that the Cauchy index in Equation (37) actually follows the classic definition (i.e., Definition 5), and is different from the one in Equation (34) (i.e., Definitions 2 and 3). Subtle differences between these two formulations have already been discussed in §4.3. Luckily, Eisermann [6] has also described an alternative sign variation operator so that our current definition of the Cauchy index (i.e., Definition 2) can be computationally evaluated:

Lemma 12 (*cindex_polyE_changes_alt_itv_mods*)

```
fixes a b::real and p q::"real poly"
assumes "a < b" and "coprime p q"
shows "cindex_polyE a b q p = changes_alt_itv_smods a b p q / 2"
```

Here, `cindex_polyE a b q p` encodes our current definition of the Cauchy index $\mathrm{Ind}_a^b(\lambda t.\, q(t)/p(t))$, and `changes_alt_itv_smods a b p q` stands for

$$\widehat{\mathrm{Var}}(\mathrm{SRemS}(p, q); a, b) \tag{38}$$

where the alternative sign variation operator $\widehat{\text{Var}}$ is defined as follows:

$$\begin{aligned}
\widehat{\text{Var}}([p_1, p_2, ..., p_3]; a, b) &= \widehat{\text{Var}}([p_1, p_2, ..., p_3]; a) - \widehat{\text{Var}}([p_1, p_2, ..., p_3]; b),\\
\widehat{\text{Var}}([p_1, p_2, ..., p_3]; a) &= \widehat{\text{Var}}([p_1(a), p_2(a), ..., p_3(a)]),\\
\widehat{\text{Var}}([]) &= 0,\\
\widehat{\text{Var}}([x_1]) &= 0,\\
\widehat{\text{Var}}([x_1, x_2, ..., x_n]) &= |\text{sgn}(x_1) - \text{sgn}(x_2)| + \widehat{\text{Var}}([x_2, ..., x_n]).
\end{aligned}$$

The difference between $\widehat{\text{Var}}$ and Var is that Var discards zeros before calculating variations while $\widehat{\text{Var}}$ takes zeros into consideration. For example, $\text{Var}([1, 0, -2]) = \text{Var}([1, -2]) = 1$, while $\widehat{\text{Var}}([1, 0, -2]) = 2$.

Before implementing Equation (34), we need to realise that there is a restriction in our strategy: roots are not allowed on the border (i.e., the image of the path $L_1+L_2+L_3+L_4$). To computationally check this restriction, the following function is defined

```
definition no_proots_line::"complex poly ⇒ complex ⇒ complex ⇒ bool" where
  "no_proots_line p a b = (proots_within p (closed_segment a b) = {})"
```

which will return "true" if there is no root on the closed segment between `a` and `b`, and "false" otherwise. Here, `closed_segment a b` is defined as the set $\{(1-u)a+ub \mid 0 \leq u \leq 1\} \subseteq \mathbb{C}$, and the function `proots_within p s` gives the set of roots of the polynomial `p` within the set `s`:

```
definition proots_within::"'a::comm_semiring_0 poly ⇒ 'a set ⇒ 'a set" where
  "proots_within p s = {x∈s. poly p x=0}"
```

The next step is to make the definition `no_proots_line` executable. This is achieved by proving a *code equation*, where the left-hand side of the equation is the target definition and the right-hand side is an executable expression. In the case of `no_proots_line`, the code equation is the following lemma:

Lemma 13 (`no_proots_line_code[code]`)

```
"no_proots_line p a b = (if poly p a ≠ 0 ∧ poly p b ≠ 0 then
                          (let p_c = p ∘_p [:a, b - a:];
                               p_R = map_poly Re p_c;
                               p_I = map_poly Im p_c;
                               g   = gcd p_R p_I
                           in if changes_itv_smods 0 1 g (pderiv g) = 0
                              then True else False)
                        else False)"
```

where $\circ_p$ is the polynomial composition operation and `map_poly Re` and `map_poly Im`, respectively, extract the real and imaginary parts of the complex polynomial `p_c`.

Proof of Lemma 13. Supposing $L : [0, 1] \to \mathbb{C}$ is a linear path from a to b: $L(t) = (1 - t)a + tb$, we know that $p \circ L$ is still a polynomial with complex coefficients. Subsequently, we extract the real and imaginary parts (p_R and p_I, respectively) of $p \circ L$ such that

$$p(L(t)) = p_R(t) + ip_I(t).$$

If there is a root of p lying right on L, we will be able to obtain some $t_0 \in [0, 1]$ such that

$$p_R(t_0) = p_I(t_0) = 0,$$

hence, by letting $g = \gcd(p_R, p_I)$ we have $g(t_0) = 0$. Therefore, the polynomial p has no (complex) root on L if and only if g has no (real) root within the interval $[0, 1]$, and the latter can be computationally checked using Sturm's theorem. □

Finally, we define the function *proots_rectangle* that returns the number of complex roots of a polynomial (counted with multiplicity) within a rectangle defined by its lower left and upper right corner:

```
definition  proots_rectangle::"complex poly ⇒ complex ⇒ complex ⇒ int" where
  "proots_rectangle p a₁ a₃ = proots_count p (box a₁ a₃)"
```

where *proots_count p s* denotes the number of roots of the polynomial p within the set s:

```
definition  proots_count::"'a::idom poly ⇒ 'a set ⇒ nat" where
  "proots_count p s = (∑ r∈proots_within p s. order r p)"
```

The executability of the function *proots_rectangle* can be established with the following code equation:

Lemma 14 (*proots_rectangle_code1[code]*)

```
"proots_rectangle p a₁ a₃ =
  (if Re a₁ < Re a₃ ∧ Im a₁ < Im a₃ then
      if p≠0 then
      if no_proots_line p a₁ (Complex (Re a₃) (Im a₁))
        ∧ no_proots_line p (Complex (Re a₃) (Im a₁)) a₃
        ∧ no_proots_line p a₃ (Complex (Re a₁) (Im a₃))
        ∧ no_proots_line p (Complex (Re a₁) (Im a₃)) a₁ then
      (
      let p₁ = p ∘ₚ [:a₁, Complex (Re a₃ - Re a₁) 0:];
          p_R1 = map_poly Re p₁; p_I1 = map_poly Im p₁; g₁ = gcd p_R1 p_I1;
          p₂ = p ∘ₚ [:Complex (Re a₃) (Im a₁), Complex 0 (Im a₃ - Im a₁):];
          p_R2 = map_poly Re p₂; p_I2 = map_poly Im p₂; g₂ = gcd p_R2 p_I2;
          p₃ = p ∘ₚ [:a₃, Complex (Re a₁ - Re a₃) 0:];
          p_R3 = map_poly Re p₃; p_I3 = map_poly Im p₃; g₃ = gcd p_R3 p_I3;
          p₄ = p ∘ₚ [:Complex (Re a₁) (Im a₃), Complex 0 (Im a₁ - Im a₃):];
          p_R4 = map_poly Re p₄; p_I4 = map_poly Im p₄; g₄ = gcd p_R4 p_I4
      in
        - (changes_alt_itv_smods 0 1 (p_R1 div g₁) (p_I1 div g₁)
          + changes_alt_itv_smods 0 1 (p_R2 div g₂) (p_I2 div g₂)
          + changes_alt_itv_smods 0 1 (p_R3 div g₃) (p_I3 div g₃)
          + changes_alt_itv_smods 0 1 (p_R4 div g₄) (p_I4 div g₄)) div 4
      )
      else Code.abort (STR ''proots_rectangle fails when there is
              a root on the border.'') (λ_. proots_rectangle p a₁ a₃)
      else Code.abort (STR ''proots_rectangle fails when p=0.'')
              (λ_. proots_rectangle p a₁ a₃)
      else 0
  )"
```

The proof of the above code equation roughly follows Equations (34) and (37), where *no_proots_line* checks if there is a root of p on the rectangle's border. Note that the gcd calculations here, such as *g₁ = gcd p_R1 p_I1*, are due to the coprime assumption in Lemma 12.

Example 5 Given a rectangle defined by $(-1, 2 + 2i)$ (as illustrated in Fig. 8) and a polynomial p with complex coefficients:

$$p(x) = x^2 - 2ix - 1 = (x - i)^2$$

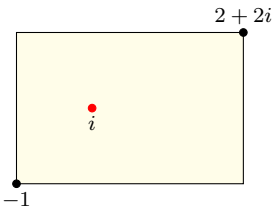

Fig. 8: A complex point i and a rectangle defined by its lower left corner -1 and upper right corner $2+2i$

we can now type the following command to count the number of roots within the rectangle:

```
value "proots_rectangle [:-1, -2*i, 1:] (-i) (2+2*i)"
```

which will return 2 as p has exactly two complex roots (i.e. i with multiplicity 2) in the area.

5.2 Roots in a Half-plane

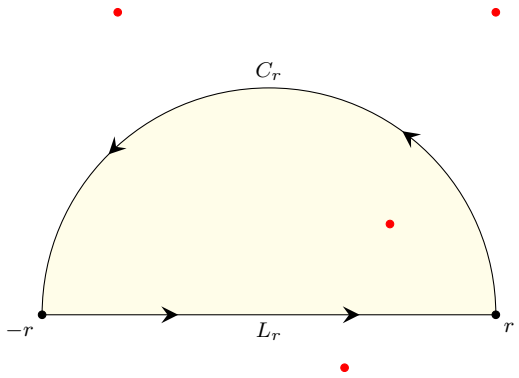

Fig. 9: Complex roots of a polynomial (red dots) and a linear path (L_r) concatenated by a semi-circular path (C_r) on the complex plane

For roots in a half-plane, we can start with a simplified case, where we count the number of roots of a polynomial in the upper half-plane of $\mathbb{C}$:

```
definition proots_upper::"complex poly ⇒ int" where
  "proots_upper p = proots_count p {z. Im z>0}"
```

As usual, our next step is to set up the executability of *proots_upper*. To achieve that, we first define a linear path $L_r(t) = (1-t)(-r)+tr$ and a semi-circular path $C_r(t) = re^{i\pi t}$, as illustrated in Fig. 9. Subsequently, let

$$\begin{aligned} C_p(r) &= p \circ C_r \\ L_p(r) &= p \circ L_r, \end{aligned}$$

and by following Equation (33) we have

$$\begin{aligned} N_r &= n(p \circ (L_r + C_r), 0) \\ &= \mathrm{Re}(n(L_p(r), 0)) + \mathrm{Re}(n(C_p(r), 0)) \end{aligned} \tag{39}$$

where N_r is the number of roots of p inside the path $L_r + C_r$. Note that as r approaches positive infinity, N_r will be the roots on the upper half-plane (i.e., *proots_upper p*), which is what we are aiming for. For this reason, it is natural for us to examine two cases:

$$\lim_{r \to +\infty} \mathrm{Re}(n(L_p(r), 0)) = ?$$

$$\lim_{r \to +\infty} \mathrm{Re}(n(C_p(r), 0)) = ?.$$

For the case of $\lim_{r\to+\infty} \mathrm{Re}(n(L_p(r), 0))$, we can have

Lemma 15 (*Re_winding_number_poly_linepth*)
fixes *p::"complex poly"*
defines *"L_p ≡ (λr::real. poly p o linepath (-r) r)"*
assumes *"lead_coeff p=1"* **and** *"∀x∈{x. poly p x=0}. Im x≠0"*
shows *"((λr. 2*Re (winding_number (L_p r) 0) + cindex_pathE (L_p r) 0)*
⟶ 0) at_top"

which essentially indicates

$$\lim_{r \to +\infty} \mathrm{Re}(n(L_p(r), 0)) = -\frac{1}{2} \lim_{r \to +\infty} \mathrm{Indp}(L_p(r), 0), \tag{40}$$

provided that the polynomial p is monic and does not have any root on the real axis.

Next, for $\lim_{r\to+\infty} \mathrm{Re}(n(C_p(r), 0))$, we first derive a lemma about C_r:

Lemma 16 (*Re_winding_number_tendsto_part_circlepath*)
fixes *z z_0::complex*
shows *"((λr. Re (winding_number (part_circlepath z r 0 pi) z_0))*
⟶ 1/2) at_top"

that is, $\lim_{r\to+\infty} \mathrm{Re}(n(C_r, 0)) = 1/2$, following which and by induction we have

Lemma 17 (*Re_winding_number_poly_part_circlepath*)
fixes *z::complex* **and** *p::"complex poly"*
defines *"C_p ≡ (λr::real. poly p o part_circlepath z r 0 pi)"*
assumes *"degree p>0"*
shows *"((λr. Re (winding_number (C_p r) 0)) ⟶ degree p/2) at_top"*

which is equivalent to

$$\lim_{r\to+\infty} \mathrm{Re}(n(C_p(r),0)) = \frac{\deg(p)}{2}, \tag{41}$$

provided $\deg(p) > 0$.

Putting Equations (40) and (41) together yields the core lemma about *proots_upper* in this section:

Lemma 18 (*proots_upper_cindex_eq*)

```
fixes p::"complex poly"
assumes "lead_coeff p=1" and "∀x∈{x. poly p x=0}. Im x≠0"
shows "proots_upper p =
        (degree p - cindex_poly_ubd (map_poly Im p) (map_poly Re p))/2"
```

where *cindex_poly_ubd (map_poly Im p) (map_poly Re p)* is mathematically interpreted as $\mathrm{Ind}_{-\infty}^{+\infty}(\lambda t.\ \mathrm{Im}(p(t))/\mathrm{Re}(p(t)))$, which is derived from $\lim_{r\to\infty} \mathrm{Indp}(L_p(r),0)$ in Equation (40) since

$$\begin{aligned}
\lim_{r\to+\infty} \mathrm{Indp}(L_p(r),0) &= \lim_{r\to+\infty} \mathrm{Indp}(L_p(r),0) \\
&= \lim_{r\to+\infty} \mathrm{Ind}_0^1 \left(\lambda t.\ \frac{\mathrm{Im}(L_p(r,t))}{\mathrm{Re}(L_p(r,t))}\right) \\
&= \lim_{r\to+\infty} \mathrm{Ind}_{-r}^{r} \left(\lambda t.\ \frac{\mathrm{Im}(p(t))}{\mathrm{Re}(p(t))}\right) \\
&= \mathrm{Ind}_{-\infty}^{+\infty} \left(\lambda t.\ \frac{\mathrm{Im}(p(t))}{\mathrm{Re}(p(t))}\right).
\end{aligned}$$

Finally, following Lemma 18, the executability of the function *proots_upper* is established:

Lemma 19 (*proots_upper_code1[code]*)

```
"proots_upper p =
    (if p ≠ 0 then
      (let p_m= smult (inverse (lead_coeff p)) p;
           p_I= map_poly Im p_m;
           p_R= map_poly Re p_m;
           g = gcd p_I p_R
       in
          if changes_R_smods g (pderiv g) = 0
          then
            (degree p - changes_R_smods p_R p_I) div 2
          else
            Code.abort (STR ''proots_upper fails when there is a root
              on the border.'') (λ_. proots_upper p)
      )
     else
        Code.abort (STR ''proots_upper fails when p=0.'')
          (λ_. proots_upper p))"
```

where

- *smult (inverse (lead_coeff p)) p* divides the polynomial *p* by its leading coefficient so that the resulting polynomial p_m is monic. This corresponds to the assumption *lead_coeff p=1* in Lemma 18.

– `changes_R_smods g (pderiv g) = 0` checks if p has no root lying on the real axis, which is due to the second assumption in Lemma 18.
– `changes_R_smods` p_R p_I evaluates

$$\text{Ind}_{-\infty}^{+\infty}\left(\lambda t.\ \frac{\text{Im}(p_I(t))}{\text{Re}(p_R(t))}\right)$$

by following Equation (37).

As for the general case of a half-plane, we can have a definition as follows:

```
definition proots_half::"complex poly ⇒ complex ⇒ complex ⇒ int" where
  "proots_half p a b = proots_count p {w. Im ((w-a) / (b-a)) > 0}"
```

which encodes the number of roots in the left half-plane of the vector $b - a$. Roots of p in this half-plane can be transformed to roots of $p \circ_p [:a, b-a:]$ in the upper half-plane of $\mathbb{C}$:

Lemma 20 (*proots_half_proots_upper*)

```
fixes a b::complex and p::"complex poly"
assumes "a≠b" and "p≠0"
shows "proots_half p a b = proots_upper (p ∘p [:a, b-a:])"
```

And so we can naturally evaluate *proots_half* through *proots_upper*:

Lemma 21 (*proots_half_code1[code]*)

```
"proots_half p a b =
    (if a≠b then
     if p≠0 then
        proots_upper (p ∘p [:a, b - a:])
     else Code.abort (STR ''proots_half fails when p=0.'')
           (λ_. proots_half p a b)
     else 0)"
```

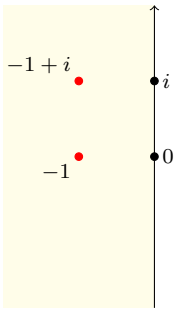

Fig. 10: Complex roots of a polynomial (red dots) and a vector $(0, i)$

Example 6 We can now use the following command

```
value "proots_half [:1-i, 2-i, 1:] 0 i"
```

to decide that the polynomial

$$p(x) = x^2 + (2 - i)x + (1 - i) = (x + 1)(x + 1 - i)$$

has exactly two roots within the left half-plane of the vector $(0, i)$, as shown in Fig. 10.

Despite our naive implementation, both `proofs_half` and `proots_rectangle` are applicable for small or medium examples. For most polynomials with coefficient bitsize up to 10 and degree up to 30, our complex root counting procedures terminate within minutes.

6 Limitations and Future Work

There are, of course, several improvements that can be made on both the evaluation tactic of §4.2 and the root counting procedures of §5. As the tactic is intended to be applied to winding numbers with variables, full automation with this tactic is unlikely in most cases, but we can always aim for better automation and an enhanced interactive experience for users (e.g., presenting unsolved goals in a more user-friendly way).

Regarding the two root-counting procedures in §5, a key limitation is that they do not allow cases where any root is on the border. There are two possible solutions to this problem:

- To generalise the definition of winding numbers. The current formulation of winding numbers in Isabelle/HOL follows the one in complex analysis:

  $$n(\gamma, z) = \frac{1}{2\pi i} \oint_\gamma \frac{dw}{w - z}$$

  which becomes undefined when the point z is on the image of the path γ. With more general formulations of winding numbers, such as the algebraic version by Eisermann [6], we may be able to derive a stronger version of the argument principle that allows zeros on the border.
- To deploy a more sophisticated strategy to count the number of times that the path winds. Recall that the underlying idea in this paper is to reduce the evaluation of winding numbers to *classifications* of how paths cross some line. The Cauchy index merely provides one classification strategy, which we considered simple and elegant enough for formalisation. In contrast, Collins and Krandick [4] propose a much more sophisticated strategy for such classifications. Their strategy has, in fact, been widely implemented in modern systems, such as Mathematica and SymPy, to count the number of complex roots.

Neither of these two solutions are straightforward to incorporate, hence we leave them for future investigation.

Besides rectangles and half-planes, it is also possible to similarly count the number of roots in an open disk and even a sector:

$$\text{sector}(z_0, \alpha, \beta) = \{z \mid \alpha < \arg(z - z_0) < \beta\}$$

where $\arg(-)$ returns the argument of a complex number. Informal proofs of root counting in these domains can be found in Rahman and Schmeisser [19, Chapter 11].

7 Potential Applications

Rahman and Schmeisser's book [19, Chapter 11] and Eisermann's paper [6] are the two main sources that our development is built upon. Nevertheless, there are still some differences in formulations:

- Rahman and Schmeisser formulated the Cauchy index as in Definitions 4 and 5, and we used their formulation in our first attempt. However, after we realised the subtleties discussed in §4.3, we abandoned this formulation and switched to Eisermann's (i.e., Definition 2). As a result, the root counting procedures presented in this paper are more general than the ones in their book, having fewer preconditions.
- Eisermann formulated a winding number $n(\gamma, z_0)$ in a real-algebraical sense where γ is required to be a piecewise polynomial path (i.e., each piece from the path needs to be a polynomial). In comparison, $n(\gamma, z_0)$ in Isabelle/HOL follows the classic definition in complex analysis, and places fewer restrictions on the shape of γ (i.e., piecewise continuously differentiable is less restrictive than being a piecewise polynomial) but does not permit z_0 to be on the image of γ (while Eisermann's formulation does). Consequently, Eisermann's root counting procedure works in more restrictive domains (i.e., he only described the rectangle case in his paper) but does not prevent roots on the border.

Another point worth mentioning is the difference between informal and formal proofs. In this development, we generally treated their lemma statements as bald facts: we had to discover our own proofs. For instance, when proving Proposition 1, we defined an inductive data type for segments and derived an induction rule for it, which was nothing like the informal proof. Such situations also happened when we justified the root counting procedure in a half-plane. Overall, the formal proofs are about 12000 lines.

Interestingly, the root-counting procedure in a half-plane is also related to the stability problems in the theory of dynamical systems. For instance, let $A \in \mathbb{R}^{n\times n}$ be a square matrix with real coefficients and $y : [0, +\infty) \to \mathbb{R}^n$ be a function that models the system state over time. A linear dynamical system can be described as an ordinary differential equation:

$$\frac{dy(t)}{dt} = Ay(t) \tag{42}$$

with an initial condition $y(0) = y_0$. The system of (42) is considered stable if all roots of the characteristic polynomial of A lie within the open left half-plane (i.e., $\{z \mid \mathrm{Re}(z) < 0\}$), and this stability test is usually referred as the Routh-Hurwitz stability criterion [1, Section 23][16, Chapter 9]. As has been demonstrated in Example 6, counting the number of roots in the left half-plane is within the scope of the procedure `proots_half`. For this reason, we believe that the development in this paper will be beneficial for reasoning about dynamical systems in Isabelle/HOL.

It is worth mentioning that root counting in a rectangle is usually coupled with a classic problem in computer algebra, namely, complex root isolation. The basic idea is to keep bisecting a rectangle (vertically or horizontally) into smaller ones until a sub-rectangle contains exactly one root or none (provided the target polynomial is square-free). Following this idea, it is possible to build a simple and verified procedure for complex root isolation similar to Wilf's algorithm [20]: we

start with a large rectangle and then repeatedly apply the verified procedure to count roots during the rectangle bisection phase. However, compared to modern complex procedures [4,21], this simplistic approach suffers from several drawbacks:

- Our root counting procedure is based on remainder sequences, which are generally considered much slower than those built upon Descartes' rule of signs.
- Modern isolation procedures are routinely required to deliver isolation boxes whose sizes meet some user-specified limit, hence they usually keep *refining* the isolation boxes even after the roots have been successfully isolated. The bisection strategy still works in the root refinement stage, but dedicated numerical approaches such as Newton's iteration are commonly implemented for efficiency reasons.
- Modern isolation procedures sometimes prefer a bit-stream model in which the coefficients of the polynomial are approximated as a bit stream. This approach is particularly beneficial when the coefficients have extremely large bit-width or consist of algebraic numbers.
- Modern implementations usually incorporate numerous low-level optimisations, such as hash tables, which are hard to implement as verified procedures in a theorem prover.

Therefore, it is unlikely that our verified root counting procedures will ever deliver high performance. Nevertheless, they can be used to certify results from untrusted external root isolation programs, as in the certificate-based approach to solving univariate polynomial problems [13].

8 Related Work

Formalisations of the winding number (from an analytical perspective) are available in Coq [3], HOL Light [7] and Isabelle/HOL. To the best of our knowledge, our tactic of evaluating winding numbers through Cauchy indices is novel. As both HOL Light and Isabelle/HOL have a relatively comprehensive library of complex analysis (i.e., at least including Cauchy's integral theorem), our evaluation tactic could be useful when deriving analytical proofs in these two proof assistants.

The ability to count the *real* roots of a polynomial only requires Sturm's theorem, so this capability is widely available among major proof assistants including PVS [18], Coq [15], HOL Light [17] and Isabelle [5,10,13]. However, as far as we know, our procedures to count *complex* roots are novel, as they require a formalisation of the argument principle [14], which is only available in Isabelle at the time of writing.

9 Conclusion

In this paper, we have described a novel tactic `winding_eval` to evaluate winding numbers via Cauchy indices: given a goal of the form

$$n(\gamma_1 + \gamma_2 + \cdots + \gamma_n, z_0) = k,$$

the tactic converts the target into an equality about Cauchy indices:

$$\mathrm{Indp}(\gamma_1, z_0) + \mathrm{Indp}(\gamma_2, z_0) + \cdots + \mathrm{Indp}(\gamma_n, z_0) = -2k.$$

This can be then solved by individually evaluating Indp(γ_1, z_0),...,Indp(γ_n, z_0). As open variables may occur in those Cauchy indices, the evaluation of them is unlikely to be fully automatic, but we provide lemmas (e.g., Lemma 11) to mitigate the laborious process. The tactic `winding_eval` has greatly helped us with the motivating proofs shown in §2, and we believe that it should be also beneficial in similar situations when dealing with winding numbers in a formal framework.

We have further related Cauchy indices to the argument principle and developed novel verified procedures to count the complex roots of a polynomial within the areas of rectangles and half-planes. Despite the limitations of not allowing roots on the border (which we will solve in future work), the ability to formally count complex roots is believed to lay the foundations for conducting stability analysis (e.g., the Routh-Hurwitz stability criterion) in the framework of the Isabelle theorem prover.

Acknowledgements. We thank Dr. Angeliki Koutsoukou-Argyraki and Anthony Bordg for commenting on the early version of this draft. The work was supported by the ERC Advanced Grant ALEXANDRIA (Project 742178), funded by the European Research Council.

References

1. Arnold, V.I.: Ordinary Differential Equations. Springer (1992)
2. Basu, S., Pollack, R., Roy, M.F.: Algorithms in Real Algebraic Geometry. Springer (2006)
3. Brunel, A.: Non-constructive complex analysis in Coq. In: N.A. Danielsson, B. Nordström (eds.) 18th International Workshop on Types for Proofs and Programs (TYPES 2011), pp. 1–15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2013)
4. Collins, G.E., Krandick, W.: An efficient algorithm for infallible polynomial complex root isolation. In: Proceedings of International Symposium on Symbolic and Algebraic Computation, ISSAC '92, pp. 189–194. ACM (1992)
5. Eberl, M.: A decision procedure for univariate real polynomials in Isabelle/HOL. In: Conference on Certified Programs and Proofs, pp. 75–83. ACM Press (2015)
6. Eisermann, M.: The Fundamental Theorem of Algebra Made Effective: An Elementary Real-algebraic Proof via Sturm Chains. The American Mathematical Monthly **119**(9), 715 (2012)
7. Harrison, J.: Formalizing basic complex analysis. In: R. Matuszewski, A. Zalewska (eds.) From Insight to Proof: Festschrift in Honour of Andrzej Trybulec, *Studies in Logic, Grammar and Rhetoric*, vol. 10(23), pp. 151–165. University of Białystok (2007). URL `http://mizar.org/trybulec65/`
8. Harrison, J.: Formalizing an analytic proof of the Prime Number Theorem (Dedicated to Mike Gordon on the occasion of his 60th birthday). Journal of Automated Reasoning **43**, 243–261 (2009)
9. Hölzl, J., Immler, F., Huffman, B.: Type classes and filters for mathematical analysis in Isabelle/HOL **7998**, 279–294 (2013). DOI 10.1007/978-3-642-39634-2_21
10. Li, W.: The Sturm-Tarski Theorem. Archive of Formal Proofs (2014). `http://isa-afp.org/entries/Sturm_Tarski.html`, Formal proof development
11. Li, W.: Count the Number of Complex Roots. Archive of Formal Proofs (2017). `http://isa-afp.org/entries/Count_Complex_Roots.html`, Formal proof development
12. Li, W.: Evaluate Winding Numbers through Cauchy Indices. Archive of Formal Proofs (2017). `http://isa-afp.org/entries/Winding_Number_Eval.html`, Formal proof development
13. Li, W., Passmore, G.O., Paulson, L.C.: Deciding Univariate Polynomial Problems Using Untrusted Certificates in Isabelle/HOL. Journal of Automated Reasoning (2017). Online at `https://link.springer.com/article/10.1007/s10817-017-9424-6`

14. Li, W., Paulson, L.C.: A formal proof of Cauchy's residue theorem. In: J.C. Blanchette, S. Merz (eds.) 7th International Conference on Interactive Theorem Proving, pp. 235–251. Springer (2016)
15. Mahboubi, A., Cohen, C.: Formal proofs in real algebraic geometry: from ordered fields to quantifier elimination. Logical Methods in Computer Science **8**(1) (2012)
16. Marden, M.: Geometry of Polynomials. Second Edition. American Mathematical Society, Providence, Rhode Island (1949)
17. McLaughlin, S., Harrison, J.: A proof-producing decision procedure for real arithmetic. In: R. Nieuwenhuis (ed.) Automated Deduction – CADE-20, pp. 295–314. Springer (2005)
18. Narkawicz, A., Muñoz, C.A., Dutle, A.: Formally-Verified Decision Procedures for Univariate Polynomial Computation Based on Sturm's and Tarski's Theorems. Journal of Automated Reasoning **54**(4), 285–326 (2015)
19. Rahman, Q.I., Schmeisser, G.: Analytic Theory of Polynomials(2002). Oxford University Press (2016)
20. Wilf, H.S.: A Global Bisection Algorithm for Computing the Zeros of Polynomials in the Complex Plane. Journal of the ACM **25**(3), 415–420 (1978)
21. Yap, C.K., Sagraloff, M.: A simple but exact and efficient algorithm for complex root isolation. In: 36th International Symposium on Symbolic and Algebraic Computation, ISSAC '11, pp. 353–360. ACM Press (2011)