Портирование **Android-приложений** в экосистему **HarmonyOS NEXT** с использованием формальных спецификаций

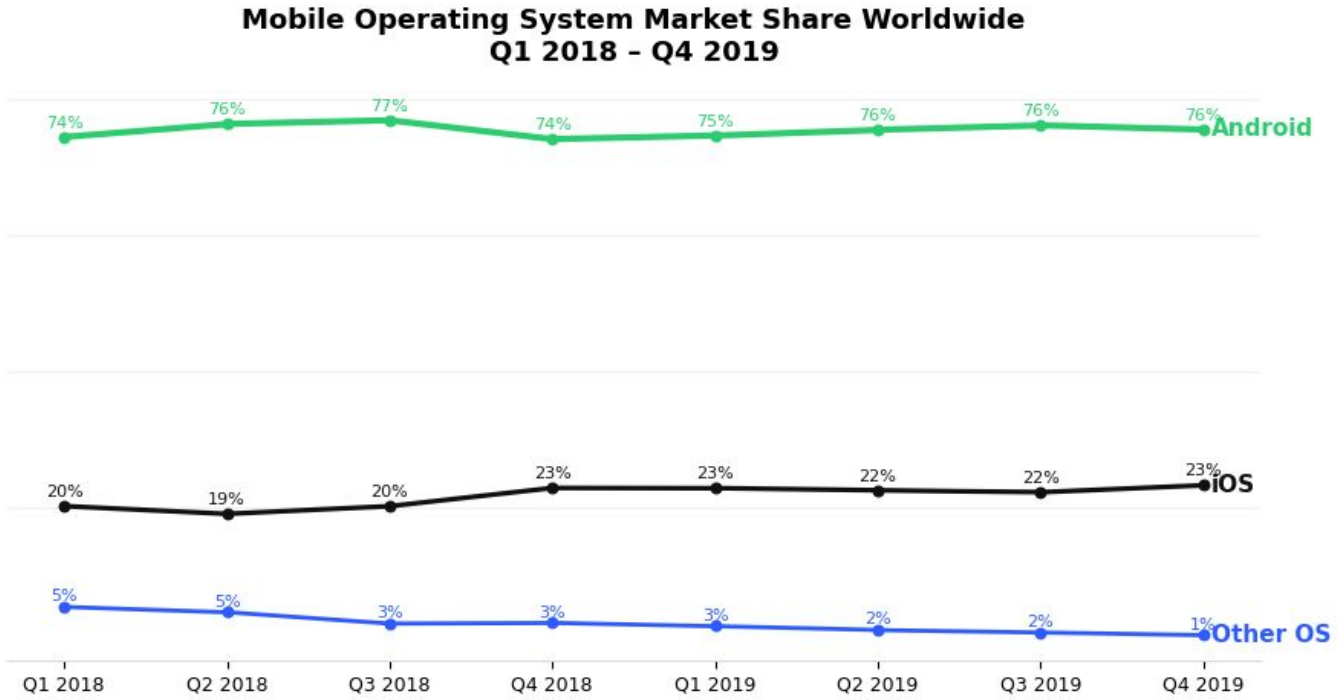
Миленин Иван | Мобильный разработчик

липтсофт

ИТМО
ИНСТИТУТ ПРИКЛАДНЫХ КОМПЬЮТЕРНЫХ НАУК

Оглавление

Мобильные ОС (2018-2019)



HarmonyOS 1.0 (First Gen.)

- Первый публичный релиз в 2019 году
- Использовалась лишь для TV и IoT
- Базировалась на **LiteOS**

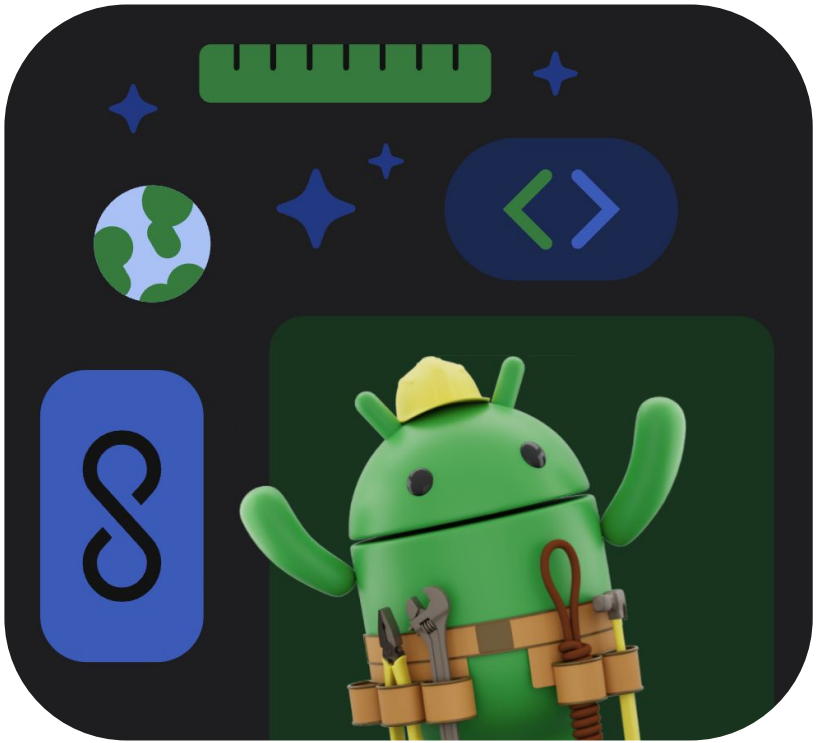
Name	OS version	API version	Initial stable release date	Latest security patch version (release date)[5]	AOSP version
HarmonyOS 1.0	1.0.0	5	August 9, 2019	1.0.1.66 SP3 (October 20, 2020)	Android 9 "Pie" (TV)
Legend: Unsupported	Supported	**Latest version**	Preview version		

HarmonyOS 2/3/4 (Second Gen.)

- Первый публичный релиз в 2021 году
- ОС были доступно для использование на смартфонах
- Базировалась на **AOSP**
- Общее число доступных приложений - **3 500 000+**

AOSP

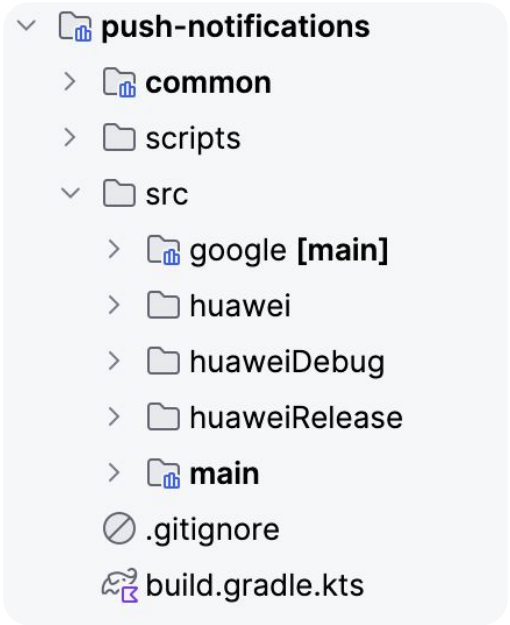
AOSP (Android Open Source Project) — открытая версия Android **без фирменных сервисов Google**.



HMS Core

HMS (Huawei Mobile Services) — альтернатива GMS (Google Mobile Services), включающая в себя:

- Huawei ID
- Push Kit
- Analytics Kit
- Map Kit
- и т.д



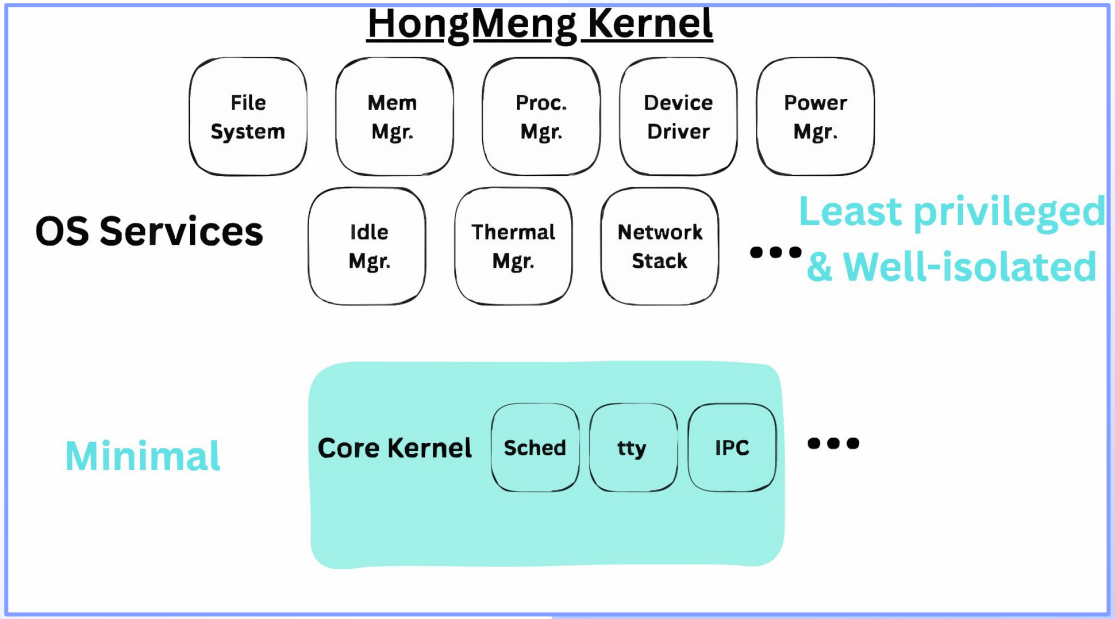
HongMeng Kernel

1

HongMeng Kernel — микроядерная архитектура, разработанная с целью заменить **AOSP** и **Linux-ядро**

2

Представлено в августе **2023** года



Новые языки и фреймворки

1

ArkTS — язык программирований, основанный на синтаксисе TypeScript

2

3

ArkTS

Новые языки и фреймворки

1. **ArkTS** — язык программирований, основанный на синтаксисе TypeScript
2. **ArkUI** — декларативный UI фреймворк
3.

ArkTS

ArkUI

Новые языки и фреймворки

1

ArkTS — язык программирований, основанный на синтаксисе TypeScript

2

ArkUI — декларативный UI фреймворк

3

DevEco — IDE для разработки прложений под HarmonyOS.

IntelliJ IDEA's fork

ArkTS

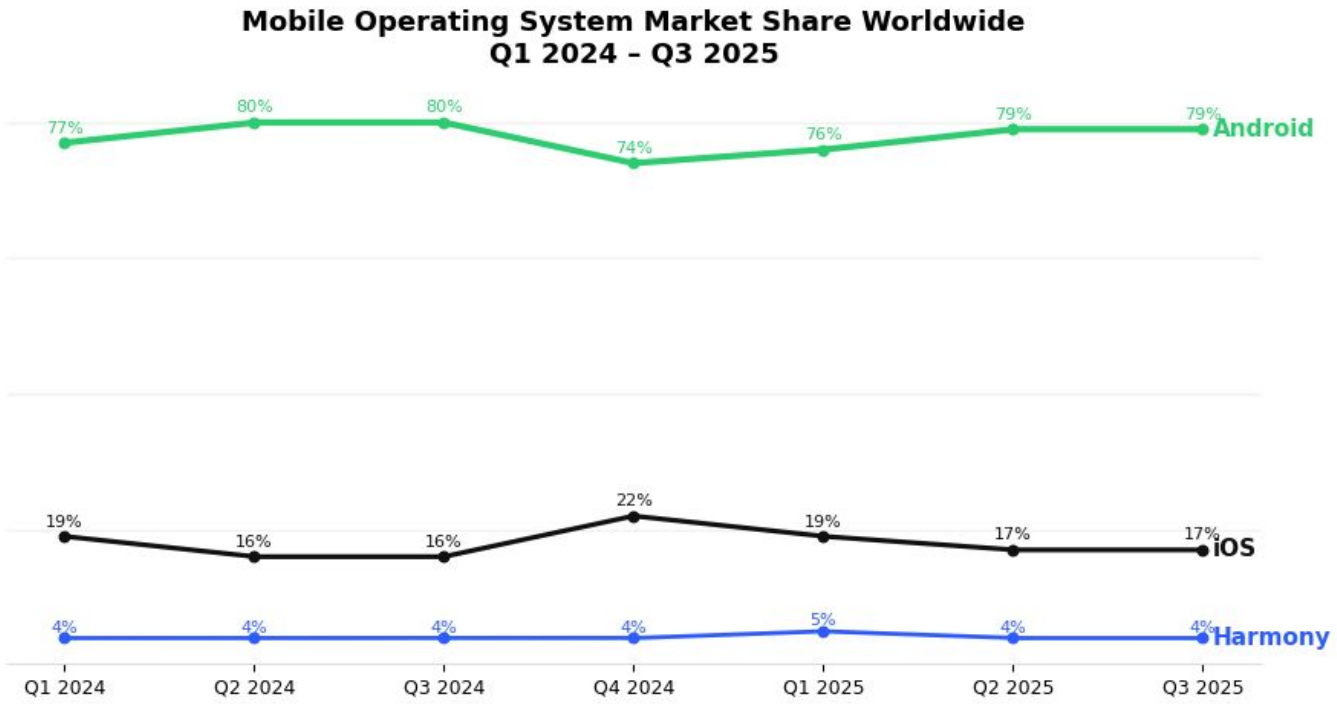
ArkUI

DevEco

HarmonyOS 5/6 (Third Gen.)

- Первый публичный релиз в 2024 году
- Базировалась на **HongMeng Kernel**
- Использует новые фреймворки и языки (**ArkTS**, **ArkUI**, **Canjie**)
- Общее число доступных приложений - **20 000+**

Мобильные ОС (2024-2025)

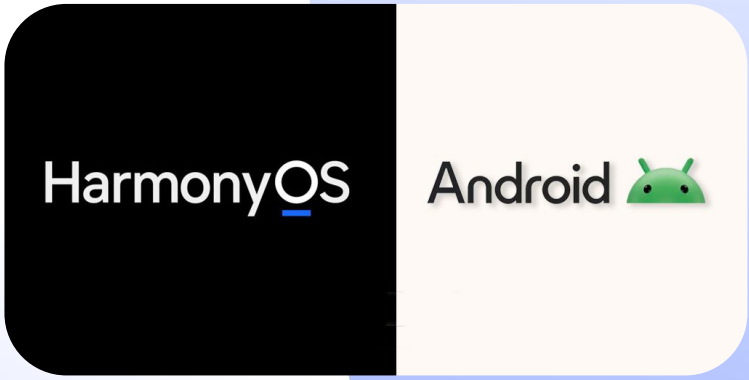


Актуальность

1. Приложений мало

2. Китайских туристов все больше и больше

Ключевая задача - **наполнение платформы HarmonyOS NEXT пользовательскими приложениями**



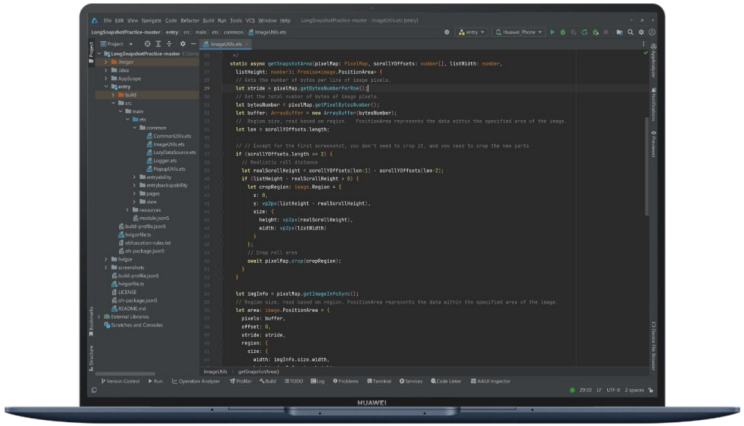
Оглавление

Существующие подходы портирования

1. Разработка с нуля
2. Кроссплатформа
3. Виртуальное окружение
4. ML/LLM
5. Автоматическая трансляция кода

Существующие подходы портирования

Разработка с нуля



— “Полный контроль” при написании кода

— Существуют официальные инструменты

— Долго

— Нужны команда разработчиков под ArkTS

Существующие подходы портирования

Кроссплатформа

— 1 код - несколько платформ

— Есть официальная поддержка **ArkUI-X**

— Нет официальной поддержки популярных инструментов

Существующие подходы портирования

Виртуальное окружение

— Затраты на адаптацию кода сведены к минимуму

卓易通

— Проблемы с производительностью

— Сложнее использовать нативное API

Существующие подходы портирования

ML/LLM

Прямой транспиляции Kotlin → ArkTS **не существует**, а ML-подходы дают лишь приблизительные результаты без гарантий корректности.

- *Gong, L., Wang, C., Cui, D., Huang, Y., Wei, M. (2024). UITrans: Seamless UI Translation from Android to HarmonyOS. Proceedings of the 16th International Conference on Internetware.*

- *Liu, R., Lin, Y., Hu, Y., Zhang, Z., Gao, X. (2024). LLM-Based Java Concurrent Program to ArkTS Converter. 2024 39th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2403-2406.*

Существующие подходы портирования

Автоматическая трансляция кода

— Сохранение семантики исходного кода

— “Проверяемость” результата портирования

— Отсутствие автоматизированного инструментария

Оглавление

В начале был Android...

Kotlin

```
fun main() {
    println("Hello, world!")
}
```

Android

липтсофт

ИТМО
ИНСТИТУТ ПРИКЛАДНЫХ КОМПЬЮТЕРНЫХ НАУК

В начале был Android...

Kotlin

```kotlin
fun main() {
    println("Hello, world!")
}
```

Jetpack Compose

```kotlin
@Composable
fun HelloWorld() {
    Text(text = "Hello, world!")
}
```

липтсофт

ИТМО
ИНСТИТУТ ПРИКЛАДНЫХ КОМПЬЮТЕРНЫХ НАУК

А потом появился HarmonyOS...

ArkTS

```
export function main(): void {
    console.log("Hello, world!")
}
```

HarmonyOS NEXT

липтсофт

ИТМО
ИНСТИТУТ ПРИКЛАДНЫХ КОМПЬЮТЕРНЫХ НАУК

А потом появился HarmonyOS...

ArkTS

```
export function main(): void {
    console.log("Hello, world!")
}
```

ArkUI

```
@ComponentV2
struct HelloWorld {
    build() {
        Text("Hello, world!")
    }
}
```

HarmonyOS NEXT

липтсофт

ИТМО ИНСТИТУТ ПРИКЛАДНЫХ КОМПЬЮТЕРНЫХ НАУК

Сравнительный анализ архитектур

Kotlin	ArkTS
`val counter: Int? = 0`	`let counter: number | null = 0;`

Сравнительный анализ архитектур

Kotlin	ArkTS
`val counter: Int? = 0`	`let counter: number | null = 0;`

`fun increment(x: Int): Int {`
`  return x + 1`
`}`
	`export function increment(x: number): number {`
	`  return x + 1;`
	`}`

Сравнительный анализ архитектур

Kotlin	ArkTS
`val counter: Int? = 0`	`let counter: number | null = 0;`

`fun increment(x: Int): Int {`
`  return x + 1`
`}`
	`export function increment(x: number): number {`
	`  return x + 1;`
	`}`

`if (x == "LIPT") {`
`  println("Cool!")`
`} else if (x == "IPKN") {`
`  println("Wow!")`
`} else {`
`  println("Awesome :)")`
`}`
	`if (x == "LIPT") {`
	`  console.log("Cool!")`
	`} else if (x == "IPKN") {`
	`  console.log("Wow!")`
	`} else {`
	`  console.log("Awesome :)")`
	`}`

Сравнительный анализ архитектур

Kotlin	ArkTS
`val counter: Int? = 0`	`let counter: number | null = 0;`

`fun increment(x: Int): Int {`
`  return x + 1`
`}`
	`export function increment(x: number): number {`
	`  return x + 1;`
	`}`

`if (x == "LIPT") {`
`  println("Cool!")`
`} else if (x == "IPKN") {`
`  println("Wow!")`
`} else {`
`  println("Awesome :)")`
`}`
	`if (x == "LIPT") {`
	`  console.log("Cool!")`
	`} else if (x == "IPKN") {`
	`  console.log("Wow!")`
	`} else {`
	`  console.log("Awesome :)")`
	`}`

Трансляция базовых конструкций

Сравнительный анализ архитектур

Kotlin	ArkTS
`val counter: Int? = 0`	`let counter: number | null = 0;`

`fun increment(x: Int): Int {`
`  return x + 1`
`}`
	`export function increment(x: number): number {`
	`  return x + 1;`
	`}`

`if (x == "LIPT") {`
`  println("Cool!")`
`} else if (x == "IPKN") {`
`  println("Wow!")`
`} else {`
`  println("Awesome :)")`
`}`
	`if (x == "LIPT") {`
	`  console.log("Cool!")`
	`} else if (x == "IPKN") {`
	`  console.log("Wow!")`
	`} else {`
	`  console.log("Awesome :)")`
	`}`

Трансляция базовых конструкций

Трансляция вызовов библиотек

Сравнительный анализ архитектур

Jetpack Compose	ArkUI

`@Composable`
`fun AnyComponent() {`
`  ...`
`}`
	`@ComponentV2`
	`struct AnyComponent {`
	`  build() {`
	`  ...`
	`  }`
	`}`

Сравнительный анализ архитектур

Jetpack Compose	ArkUI

`@Composable`
`fun AnyComponent() {`
`  ...`
`}`
	`@ComponentV2`
	`struct AnyComponent {`
	`  build() {`
	`  ...`
	`  }`
	`}`

`Button(onClick = { counter++ }) {`
`  Text("Add")`
`}`
	`Button("Add")`
	`.onClick(() => { this.counter++; });`

Сравнительный анализ архитектур

Jetpack Compose	ArkUI

`@Composable`
`fun AnyComponent() {`
`  ...`
`}`
	`@ComponentV2`
	`struct AnyComponent {`
	`  build() {`
	`  ...`
	`  }`
	`}`

`Button(onClick = { counter++ }) {`
`  Text("Add")`
`}`
	`Button("Add")`
	`.onClick(() => { this.counter++; });`

`var counter by remember { mutableStateOf(0) }`	`@State counter: number = 0;`

Сравнительный анализ архитектур

Jetpack Compose	ArkUI

`@Composable`
`fun AnyComponent() {`
`  ...`
`}`
	`@ComponentV2`
	`struct AnyComponent {`
	`  build() {`
	`  ...`
	`  }`
	`}`

`Button(onClick = { counter++ }) {`
`  Text("Add")`
`}`
	`Button("Add")`
	`.onClick(() => { this.counter++; });`

`var counter by remember { mutableStateOf(0) }`	`@State counter: number = 0;`

Трансляция UI

Предлагаемый подход

1

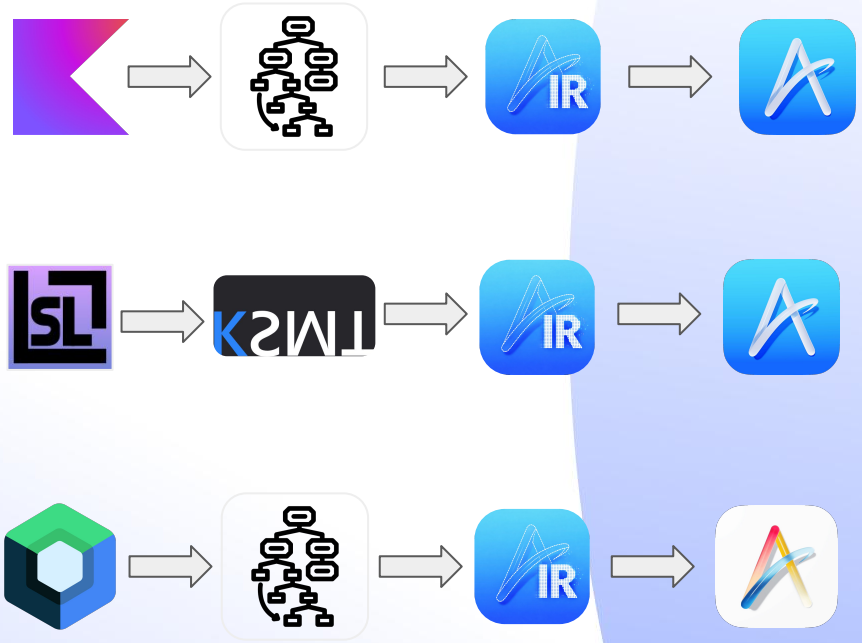
Трансляция базовых конструкций (Kotlin → ArkTS)

2

Трансляция вызовов библиотек

3

Трансляция UI

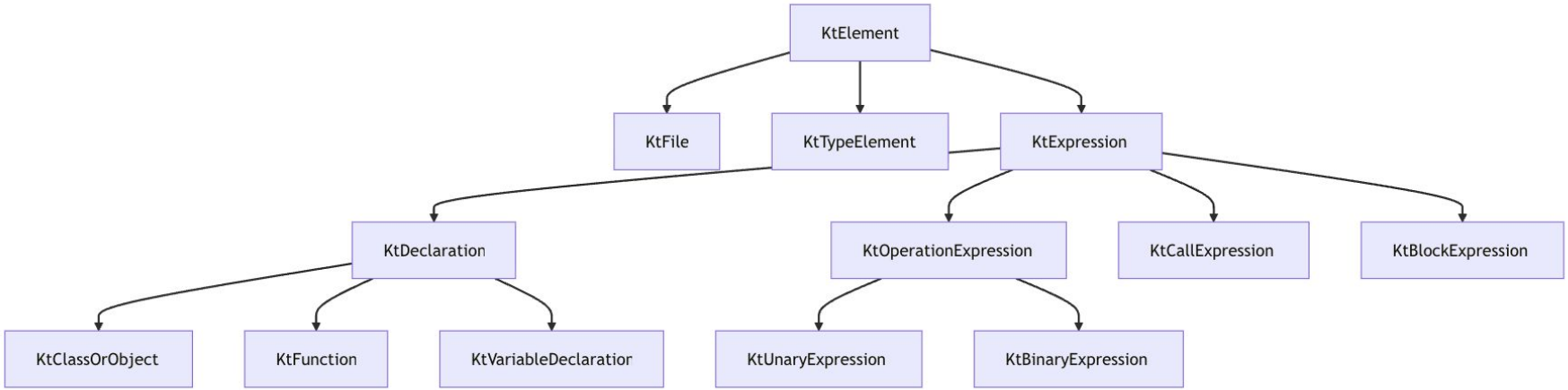


Оглавление

Kotlin PSI

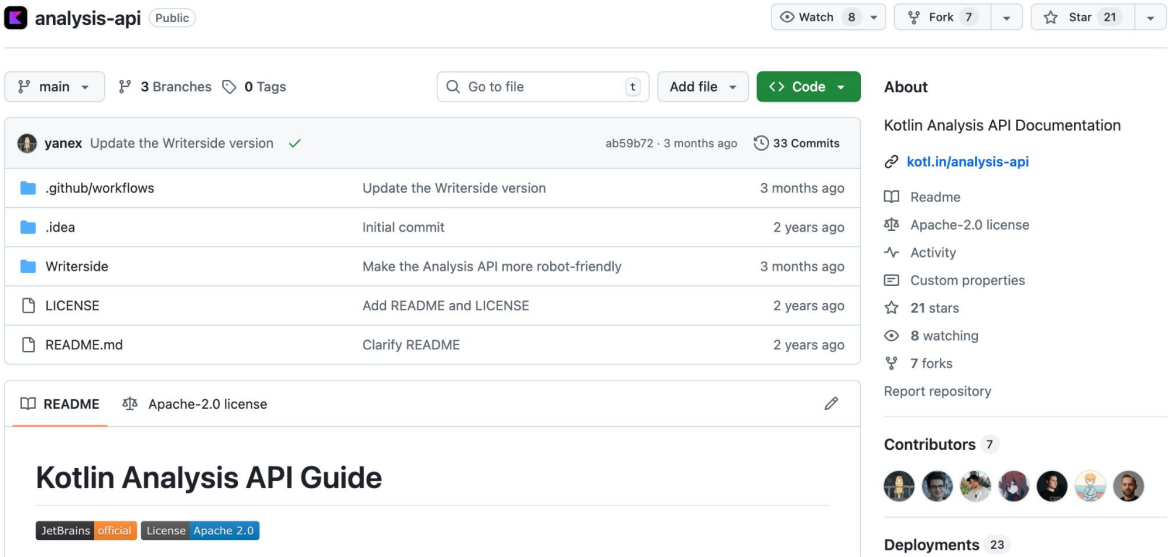
Program Structure Interface (PSI) *— это слой, отвечающий за анализ файлов и создание синтаксической и семантической модели кода, которая обеспечивает многие функции платформы.*

Простыми словами: *“Синтаксическое дерево, где* ***каждому элементу кода*** *соответствует* ***вершина графа****”*



Трансляция исходного кода. Kotlin → PSI

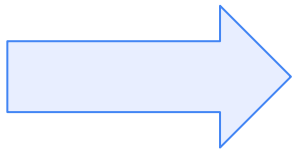
Для анализа исходного кода используется **PSI (Program Structure Interface)**, формируемый через **Kotlin Analysis API**.



Трансляция исходного кода. Kotlin → PSI

Обход PSI выполняется от корневого узла **KtFile**, далее обрабатываются классы, функции, выражения и декларации.

```
fun main() {
    val x = 10
    println(x)
}
```

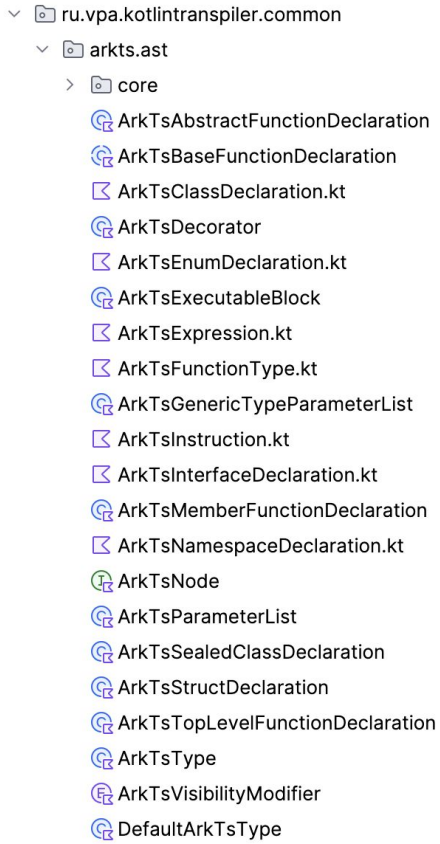


```
PSI Tree

KtFile
  KtNamedFunction (fun main() {...})
    KtModifierList
    KtIdentifier (main)
    KtParameterList (())
    KtBlockExpression ({...})
      KtProperty (val x = 10)
        KtModifierList
        KtIdentifier (x)
        KtTypeReference (null)
        KtEquals (=)
        KtConstantExpression (10)
      KtCallExpression (println(x))
```

Трансляция исходного кода. PSI → ArkTS IR

На основе PSI формируется ArkTS IR — внутреннее представление, отражающее структуру будущей ArkTS-программы.

```kotlin
/**
 * Base contract for every ArkTS AST element.
 * Each node should be able to render itself back to ArkTS source code.
 */
interface ArkTsNode {

    /**
     * Converts this AST node back to ArkTS code representation.
     */
    fun render(): String
}
```



Трансляция исходного кода. PSI → ArkTS IR

ArkTS IR **не копирует** синтаксис Kotlin, а моделирует программу в терминах **конструкций ArkTS** с учётом различий языков.

```kotlin
/**
 * Represents an identifier usage.
 *
 * Example: val a = b + 1;
 *          b - IdentifierExpression
 */
data class ArkTsIdentifierExpression(
    val name: String,
) : ArkTsExpression {

    override fun render(): String = name
}
```

```kotlin
/**
 * Represents an ArkTS expression wrapped in parentheses.
 *
 * Example: (1+2), where are all this expression is Parenthesized
 */
data class ArkTsParenthesizedExpression(
    val expression: ArkTsExpression,
) : ArkTsExpression {

    override fun render(): String {
        return buildString {
            append(ArkTsTokens.LEFT_BRACKET)
            append(expression.render())
            append(ArkTsTokens.RIGHT_BRACKET)
        }
    }
}
```

Трансляция исходного кода. PSI → ArkTS IR

ArkTS IR служит основой для генерации итогового ArkTS-кода.

```
ArkTsFile
  ArkTsTopLevelFunctionDeclaration
    name: main
    visibility: export
    returnType: ArkTsType(void)
    ArkTsParameterList
      (no parameters)
    ArkTsExecutableBlock
      ArkTsVariableDeclarationInstruction
        name: x
        isMutable: false
        initializer:
          ArkTsLiteralExpression
            value: "10"
      ArkTsExpressionInstruction
        ArkTsCallExpression
          callee:
            ArkTsIdentifierExpression
              name: println
          arguments:
            ArkTsIdentifierExpression
              name: x
```

Трансляция исходного кода. Data-классы

Kotlin

```kotlin
data class Foo(
    val a: Int,
    val b: String,
)
```

ArkTS

```typescript
export class Foo {
  public readonly a: number;

  public readonly b: string;

  constructor(a: number, b: string) {
    this.a = a
    this.b = b
  }

  public equals(other: object | null): boolean {
    return other === this || other instanceof Foo && this.a === other.a && this.b === other.b
  }

  public toString(): string {
    return "Foo(" + "a=" + this.a + ", " + "b=" + this.b + ")"
  }

  public copy(a: number = this.a, b: string = this.b): Foo {
    return new Foo(a, b)
  }
}
```

Трансляция исходного кода. Data-классы

Kotlin (.class)

```java
public final class Foo {
   private final int a;
   @NotNull
   private final String b;

   public Foo(int a, @NotNull String b) {
      Intrinsics.checkNotNullParameter(b, "b");
      super();
      this.a = a;
      this.b = b;
   }

   public final int getA() { return this.a; }

   @NotNull
   public final String getB() { return this.b; }

   public final int component1() { return this.a; }

   @NotNull
   public final String component2() { return this.b; }
```

ArkTS

```typescript
export class Foo {
  public readonly a: number;

  public readonly b: string;

  constructor(a: number, b: string) {
    this.a = a
    this.b = b
  }
```

Трансляция исходного кода. Data-классы

Kotlin (.class)

```
public boolean equals(@Nullable Object other) {
   if (this == other) {
      return true;
   } else if (!(other instanceof Foo)) {
      return false;
   } else {
      Foo var2 = (Foo)other;
      if (this.a != var2.a) {
         return false;
      } else {
         return Intrinsics.areEqual(this.b, var2.b);
      }
   }
}
```

ArkTS

```
public equals(other: object | null): boolean {
  return other === this
    || other instanceof Foo
      && this.a === other.a
      && this.b === other.b
}
```

Трансляция исходного кода. Data-классы

Kotlin (.class)

```
@NotNull
public String toString() {
   return "Foo(a=" + this.a + ", b=" + this.b + ")";
}
```

ArkTS

```
public toString(): string {
  return "Foo(" + "a=" + this.a + ", " + "b=" + this.b + ")"
}
```

Трансляция исходного кода. Data-классы

Kotlin (.class)

```java
@NotNull
public final Foo copy(int a, @NotNull String b) {
   Intrinsics.checkNotNullParameter(b, "b");
   return new Foo(a, b);
}

// $FF: synthetic method
public static Foo copy$default(Foo var0, int var1, String var2, int var3, Object var4) {
   if ((var3 & 1) != 0) {
      var1 = var0.a;
   }

   if ((var3 & 2) != 0) {
      var2 = var0.b;
   }

   return var0.copy(var1, var2);
}
```

ArkTS

```
public copy(a: number = this.a, b: string = this.b): Foo {
  return new Foo(a, b)
}
```

Трансляция исходного кода. Sealed-классы

Kotlin

```kotlin
sealed class Result {

    data class Success(
        val data: String
    ): Result()

    object Failure : Result()

}
```

ArkTS

```arkts
export abstract class Result {
}
export namespace Result {
  export class Success extends Result {
    public readonly data: string;

    constructor(data: string) {
      super()
      this.data = data
    }
  }

  export class FailureClass extends Result {
    constructor() {
      super()
    }
  }

  export const Failure: Result = new FailureClass();
}
```

Трансляция исходного кода. Лямбды

Kotlin

```kotlin
fun createLambda(): ((String) -> String) -> ((Int) -> Int) -> (Int) -> String {
    return { stringTransformer ->
        { intTransformer ->
            { value: Int ->

                val transformedInt = intTransformer(value)
                val base = "value=$value, transformed=$transformedInt"
                stringTransformer(base)

            }
        }
    }
}
```

ArkTS

```typescript
export function createLambda(): (p0: (p0: string) => string) => (p0: (p0: number) => number) => (p0: number) => string {
  return (stringTransformer) => {
    return (intTransformer) => {
      return (value) => {
        const transformedInt = intTransformer(value);
        const base = `value=${value}, transformed=${transformedInt}`;
        return stringTransformer(base)
      }
    }
  }
}
```

Трансляция исходного кода. Форматированные строки

Kotlin

```kotlin
val base = "value=$value, transformed=$transformedInt"
```

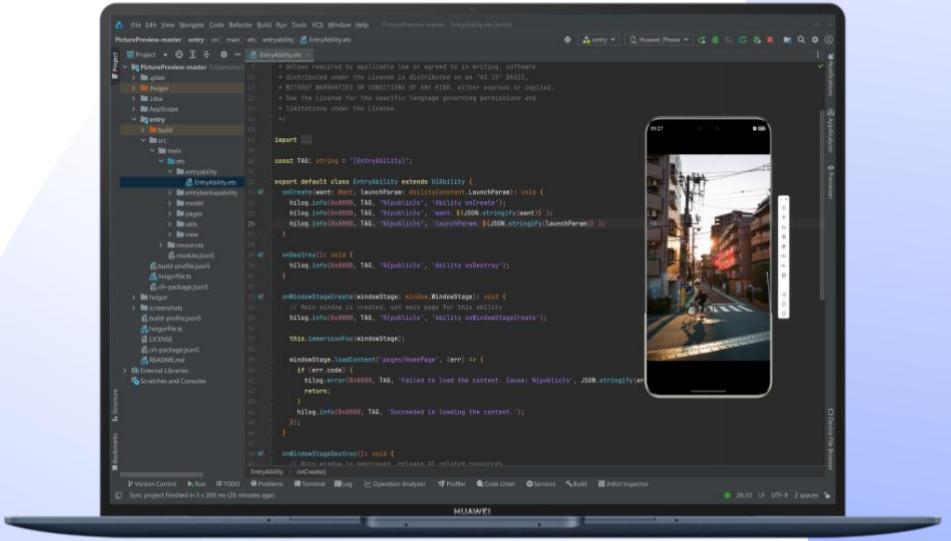
ArkTS

```
const base = `value=${value}, transformed=${transformedInt}`;
```

Оглавление

ArkUI

1. Декларативный UI фреймворк
2. Re-rendering mechanism
3. Разделение “логики” и UI



Разбор @Composable компонента

```
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Счётчик: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```

Разбор @Composable компонента

```
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Счётчик: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```

Наименование компонента

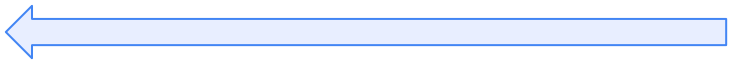
Разбор @Composable компонента

```
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Счётчик: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```

Передаваемые параметры

Разбор @Composable компонента

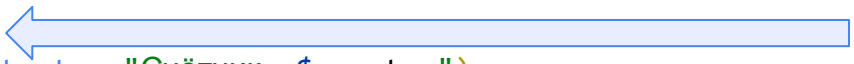
```
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Счётчик: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```



Локальный параметр

Разбор @Composable компонента

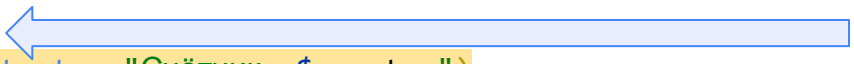
```
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Счётчик: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```



Вызов дочернего компонента

Разбор @Composable компонента

```
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Счётчик: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```



Content-лямбда

Разбор @Composable компонента

```
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Счётчик: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```

Наименование вызываемого компонента

Разбор @Composable компонента

```
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Счётчик: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```

Аргументы вызываемого компонента

Создание ArkUI компонента

```
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Счётчик: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```

Наименование компонента

```
@ComponentV2
struct CounterButton {

    build() {
    }
}
```

Создание ArkUI компонента

```
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Счётчик: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```


```
@ComponentV2
struct CounterButton {

    @Param
    @Require
    baseText: string;

    build() {
    }
}
```

Разбор @Composable компонента

```
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Счётчик: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```

Локальный параметр

```
@ComponentV2
struct CounterButton {

    @Param
    @Require
    baseText: string;

    @Local
    counter: number = 0

    build() {
    }
}
```

Разбор @Composable компонента

```
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Счётчик: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```

Вызов дочернего компонента

```
@ComponentV2
struct CounterButton {

    @Param
    @Require
    baseText: string;

    @Local
    counter: number = 0

    build() {
        Column()
    }
}
```

Разбор @Composable компонента

```kotlin
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Счётчик: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```

Content-лямбда

```kotlin
@ComponentV2
struct CounterButton {

    @Param
    @Require
    baseText: string;

    @Local
    counter: number = 0

    build() {
        Column() {
        }
    }
}
```

Разбор @Composable компонента

```
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Счётчик: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```

Вызов дочернего компонента (1)

```
@ComponentV2
struct CounterButton {

    @Param
    @Require
    baseText: string;

    @Local
    counter: number = 0

    build() {
        Column() {
            Text("Счётчик: " + this.counter)
        }
    }
}
```

Разбор @Composable компонента

```
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Счётчик: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```

Вызов дочернего компонента (2)

```
@ComponentV2
struct CounterButton {

    @Param
    @Require
    baseText: string;

    @Local
    counter: number = 0

    build() {
        Column() {
            Text("Счётчик: " + this.counter)
            Button() {
              Text(this.baseText + ": " + this.counter)
              }.onClick(() => {
                 this.counter++
              })
        }
    }
}
```

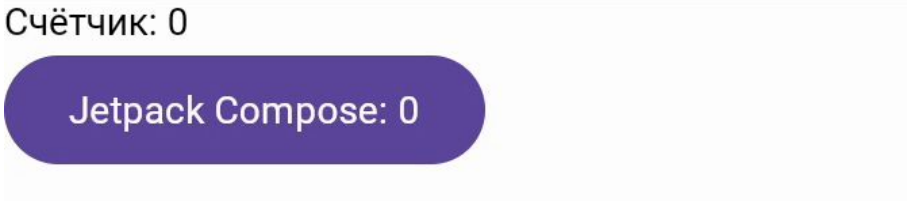
Итог портирования компонента

```
@Composable
fun CounterButton(
    baseText: String
) {
    var counter = 0

    Column {
        Text(text = "Counter: $counter")
        Button(
            onClick = { counter++ }
        ) {
            Text(text = "$baseText: $counter")
        }
    }
}
```

```
@ComponentV2
struct CounterButton {

    @Param
    @Require
    baseText: string;

    @Local
    counter: number = 0

    build() {
        Column() {
            Text("Counter: " + this.counter)
            Button() {
              Text(`${this.baseText}: ${this.counter)}`)
              .onClick(() => {
                 this.counter++
              })
        }
    }
}
```

Итог портирования компонента

Jetpack Compose



ArkUI

Сопоставление компонентов

Params: `modifier` - The modifier to be applied to the Column.
`verticalArrangement` - The vertical arrangement of the layout's children.
`horizontalAlignment` - The horizontal alignment of the layout's children.
`content` - The content of the Column

See Also: `Row`, `androidx.compose.foundation.lazy.LazyColumn`

Samples: `androidx.compose.foundation.layout.samples.SimpleColumn`

```
// Unresolved
```

```
@Composable
inline fun Column(
    modifier: Modifier = Modifier,
    verticalArrangement: Arrangement.Vertical = Arrangement.Top,
    horizontalAlignment: Alignment.Horizontal = Alignment.Start,
    content: @Composable ColumnScope.() -> Unit,
) {...}
```

Jetpack Compose

Column

Supported devices: Phone | PC/2in1 | Tablet | TV | Wearable

Column(options?: ColumnOptions)

Creates a vertical linear layout container. You can set the spacing between child components, which can be of type number or string.

Widget capability: This API can be used in ArkTS widgets since API version 9.

Atomic service API: This API can be used in atomic services since API version 11.

System capability: SystemCapability.ArkUI.ArkUI.Full

Parameters

Name	Type	Mandatory	Description
options	ColumnOptions[18+]	No	Vertical spacing between two adjacent child components.

ArkUI

Сопоставление компонентов

Params: `modifier` - The modifier to be applied to the Column.
`verticalArrangement` - The vertical arrangement of the layout's children.
`horizontalAlignment` - The horizontal alignment of the layout's children.
`content` - The content of the Column

See Also: `Row`, `androidx.compose.foundation.lazy.LazyColumn`

Samples: `androidx.compose.foundation.layout.samples.SimpleColumn`

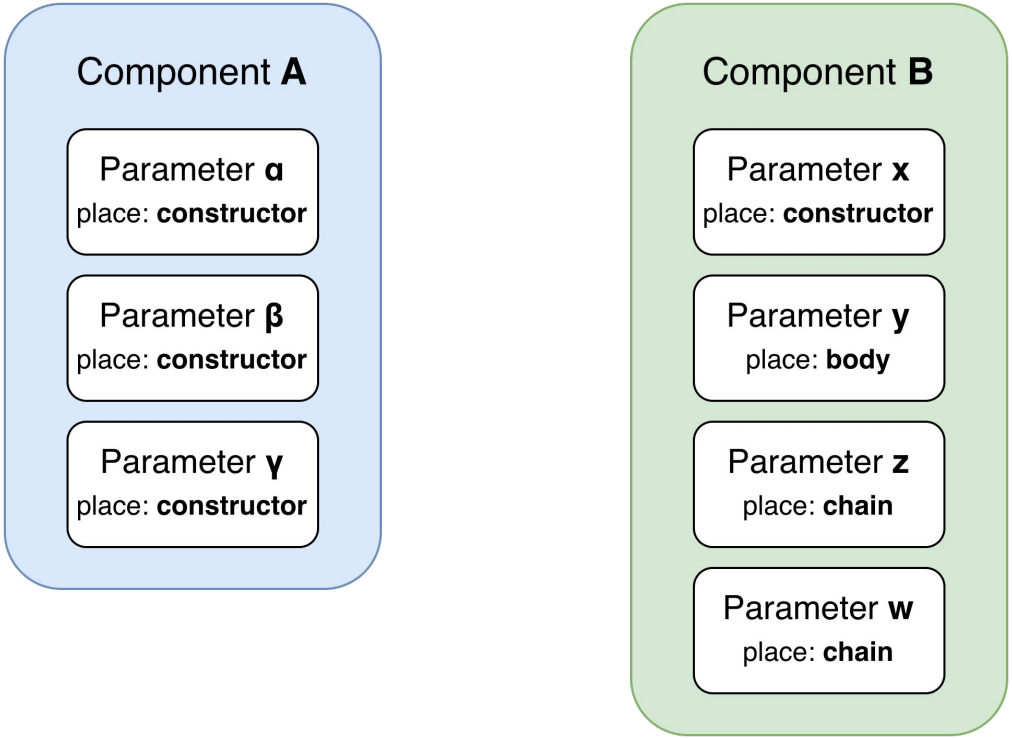
```
// Unresolved
```

```
@Composable
inline fun Column(
    modifier: Modifier = Modifier,
    verticalArrangement: Arrangement.Vertical = Arrangement.Top,
    horizontalAlignment: Alignment.Horizontal = Alignment.Start,
    content: @Composable ColumnScope.() -> Unit,
) {...}
```

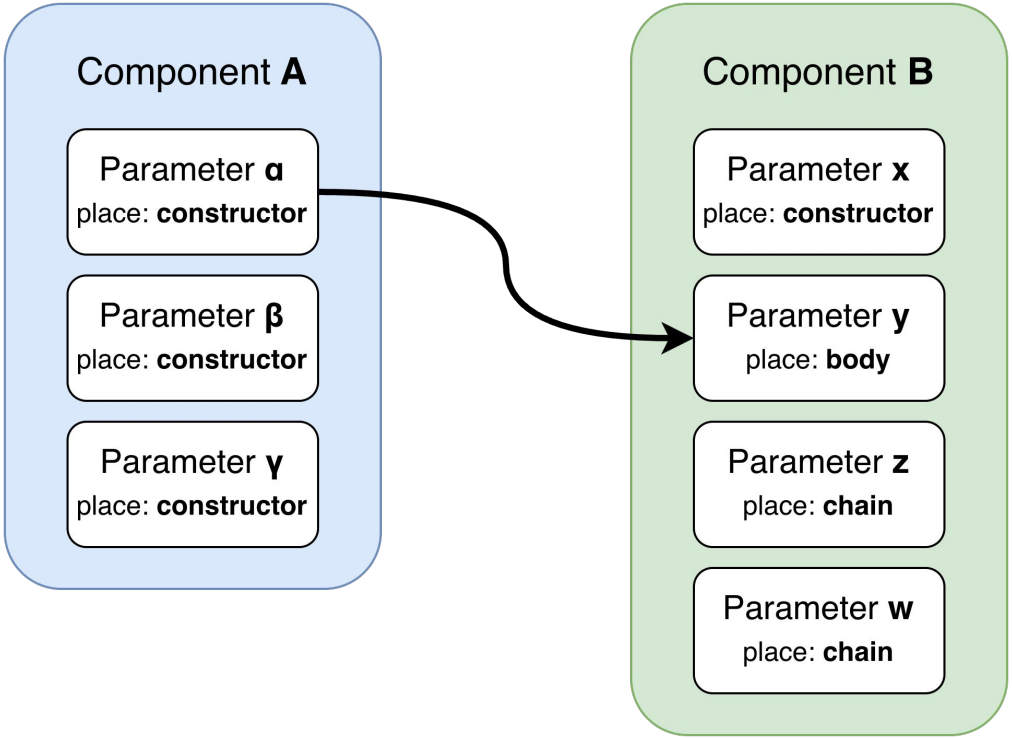
```
Column(
  /** { space: 20 } **/
) {
  /** content **/
}.justifyContent(FlexAlign.Start)
.alignItems(HorizontalAlign.Start)
```

Jetpack Compose **ArkUI**

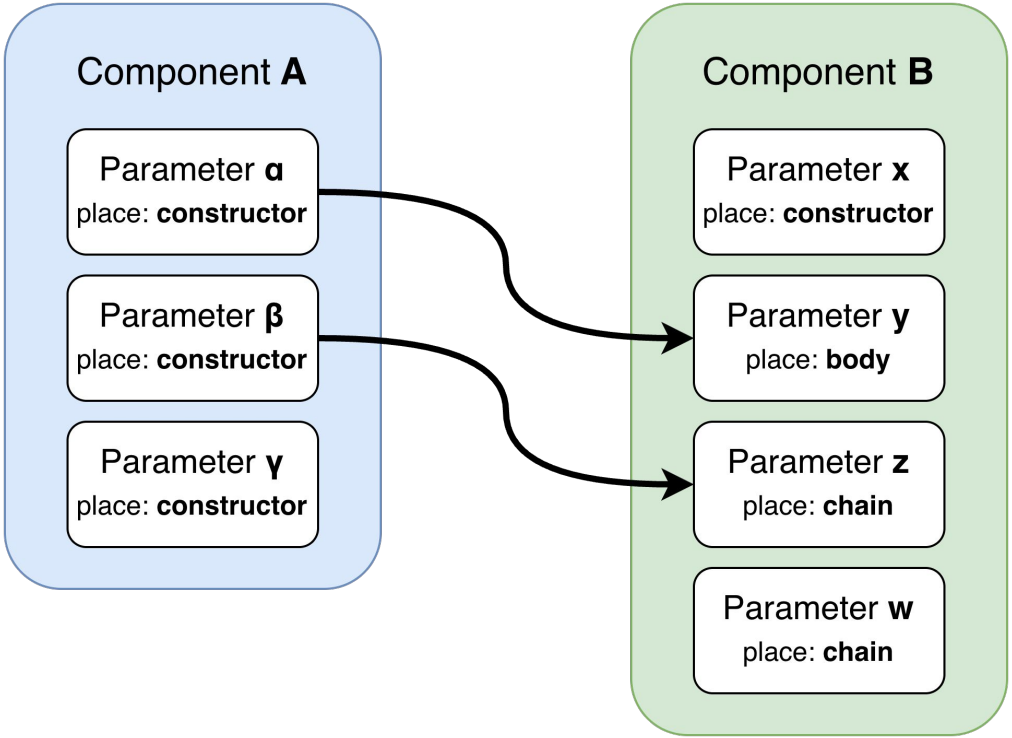
Подход сопоставления компонентов



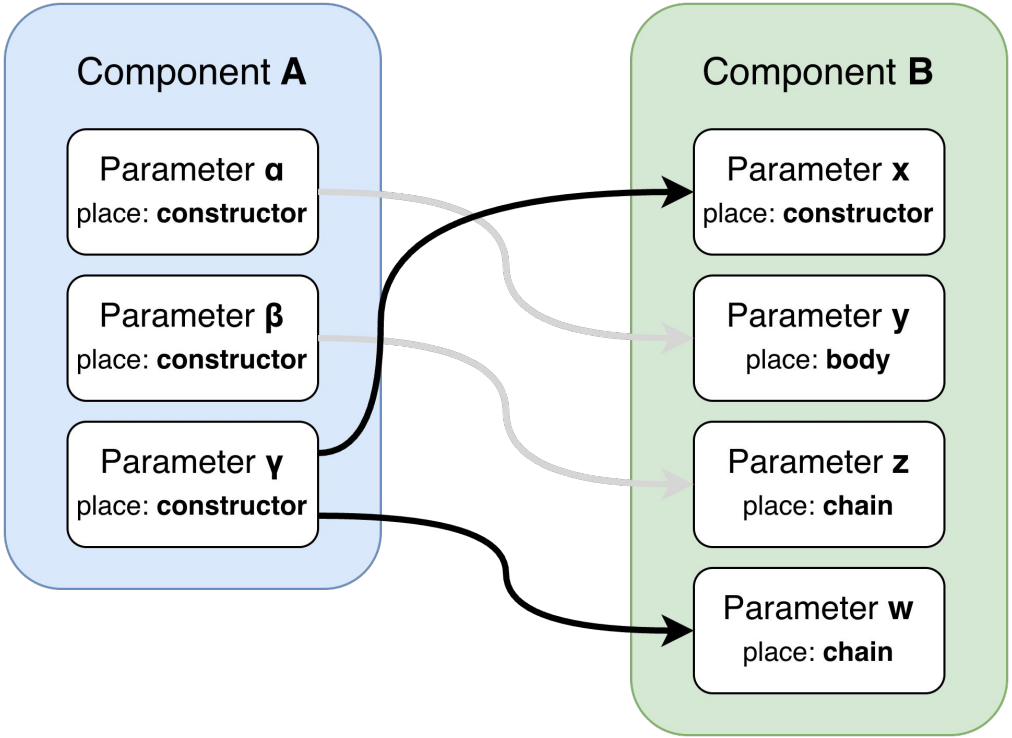
Подход сопоставления компонентов



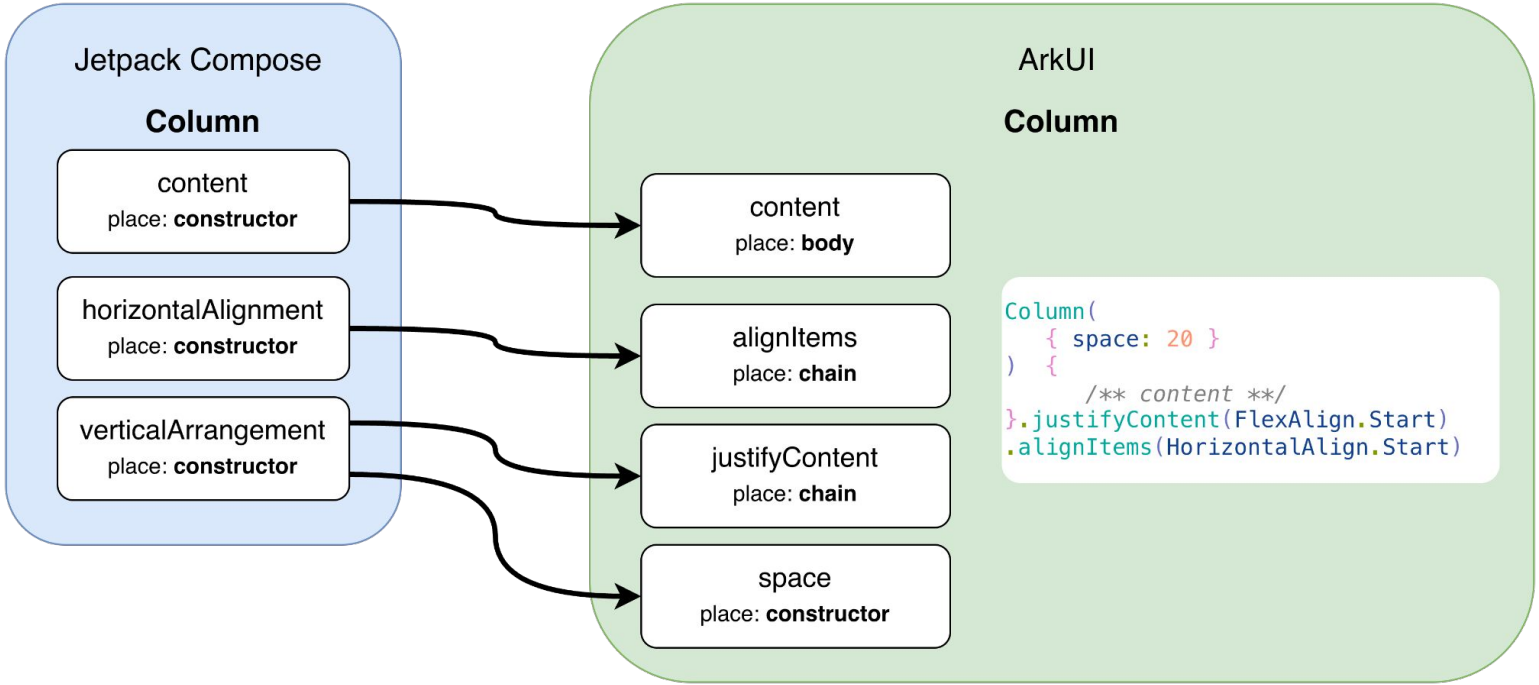
Подход сопоставления компонентов



Подход сопоставления компонентов



Подход сопоставления компонентов



Шаги добавления новых компонентов

1

Указываем наименование компонента

```
internal object TextComponentRules : ComponentRuleSetProvider {

    override val ruleSets: List<ComponentRuleSet> = listOf(
        createTextRuleSet( sourceFqName = "androidx.compose.material.Text"),
    )
```

Шаги добавления новых компонентов

2

Перечисляем все его параметры

```
private fun createTextRuleSet(sourceFqName: String): ComponentRuleSet {
    return ComponentRuleSet(
        sourceFqName = sourceFqName,
        parameterRules = listOf(
            ParameterRule(
                parameterName = "text",
                case = …,
            ),
            ParameterRule(
                parameterName = "modifier",
                case = …,
            ),
            ParameterRule(
                parameterName = "color",
                case = …,
            ),
…
```

Шаги добавления новых компонентов

3

Для каждого параметра указываем преобразование его **PSI** элемента в **ArkTS IR**

```
private fun createTextRuleSet(sourceFqName: String): ComponentRuleSet {
    return ComponentRuleSet(
        sourceFqName = sourceFqName,
        parameterRules = listOf(
            ParameterRule(
                parameterName = "text",
                case = ParameterRuleCase(
                    mutate = { input -> RuleMutationResult.single(input.mappedExpression) },
                    writes = …,
                ),
            ),
            ParameterRule(
                parameterName = "modifier",
                case = ignoredCase(),
            ),
            ParameterRule(
                parameterName = "color",
                case = ParameterRuleCase(
                    mutate = { input -> RuleMutationResult.single(input.mappedExpression) },
                    writes = …,
                ),
            ),
…
```

Шаги добавления новых компонентов

4

Указываем **стратегию** преобразования

```
parameterRules = listOf(
    ParameterRule(
        parameterName = "text",
        case = ParameterRuleCase(
            mutate = { input -> RuleMutationResult.single(input.mappedExpression) },
            writes = listOf(
                ParameterWrite(channel = TargetWriteChannel.ConstructorArgument(index = 0)),
            ),
        ),
    ),
    ParameterRule(
        parameterName = "modifier",
        case = ignoredCase(),
    ),
    ParameterRule(
        parameterName = "color",
        case = ParameterRuleCase(
            mutate = { input -> RuleMutationResult.single(input.mappedExpression) },
            writes = listOf(
                ParameterWrite(channel = TargetWriteChannel.ChainCall(methodName = CHAIN_FONT_COLOR)),
            ),
        ),
    ),
```

Пример портирования Text

```kotlin
@Composable
private fun Root() {
    Text(
        text = "Hello",
        color = Color.Red,
        fontSize = 18.sp,
        fontStyle = FontStyle.Italic,
        fontWeight = FontWeight.W600,
        letterSpacing = 1.5.sp,
        textAlign = TextAlign.Center,
        lineHeight = 22.sp,
        overflow = TextOverflow.Visible,
        maxLines = 3,
    )
}
```

Jetpack Compose

```kotlin
@ComponentV2
struct Root {
  build() {
    Text("Hello")
      .fontColor(Color.Red)
      .fontSize(18)
      .fontStyle(FontStyle.Italic)
      .fontWeight(600)
      .letterSpacing(1.5)
      .textAlign(TextAlign.Center)
      .lineHeight(22)
      .textOverflow({ overflow: TextOverflow.None })
      .maxLines(3)
  }
}
```

ArkUI

А что по Modifier?

Modifier

Foundation Component

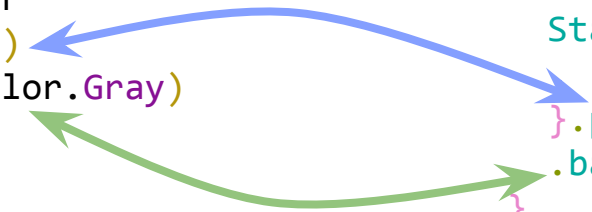
Custom Component

```
@Composable
fun Foo(
    modifier: Modifier
) {
    Text(modifier = modifier, ...)
    ...
}
```

Foundation Component's Modifier

```
@Composable
private fun MyBox() {
    Box(
        modifier = Modifier
            .padding(16.dp)
            .background(Color.Gray)
    ) {
        // content
    }
}
```

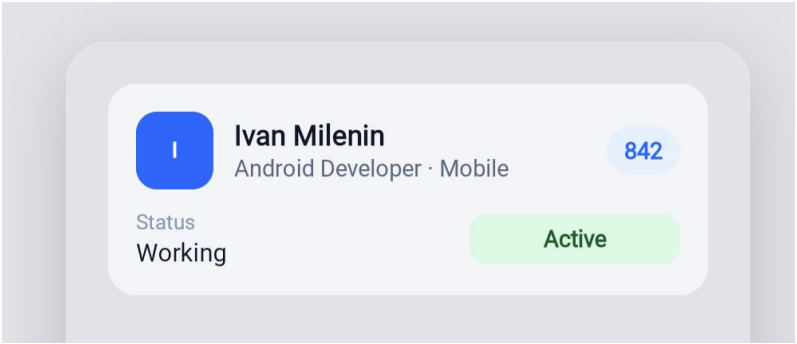
```
@ComponentV2
struct MyBox {
 build() {
   Stack() {

   }.padding(16)
   .backgroundColor(Color.Gray)
  }
}
```



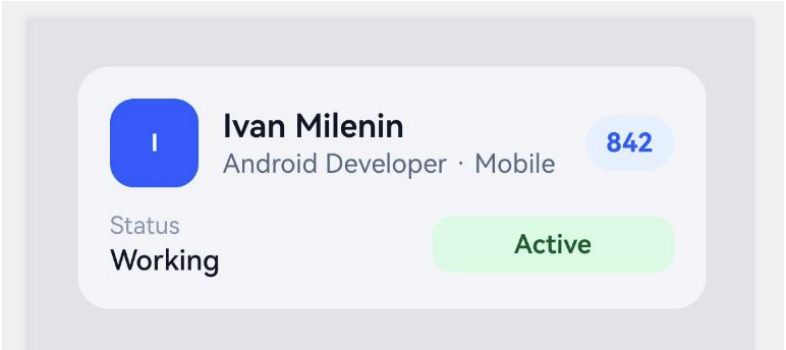
Jetpack Compose

ArkUI

Пример реального использования. Personal card



Jetpack Compose



ArkUI

Пример реального использования. Personal card

Оглавление

Трансляция вызовов библиотек

Kotlin

```kotlin
fun enterPin(
    pin: String,
) {
    val firstFourDigits = pin.take(4)
    println("PIN (first 4 digits): " + firstFourDigits)
    currentPin = pin
}
```

ArkTS

```typescript
export function enterPin(pin: string): void {
    const firstFourDigits: string = pin.substring(0, 4)
    console.log('PIN (first 4 digits): ' + firstFourDigits)
    this.currentPin = pin
}
```

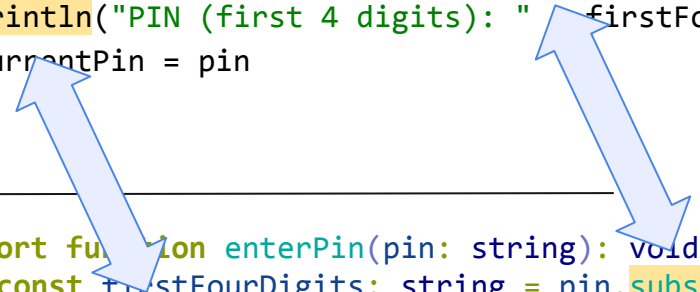
Трансляция вызовов библиотек

Kotlin

```kotlin
fun enterPin(
    pin: String,
) {
    val firstFourDigits = pin.take(4)
    println("PIN (first 4 digits): " + firstFourDigits)
    currentPin = pin
}
```

ArkTS

```typescript
export function enterPin(pin: string): void {
    const firstFourDigits: string = pin.substring(0, 4)
    console.log('PIN (first 4 digits): ' + firstFourDigits)
    this.currentPin = pin
}
```



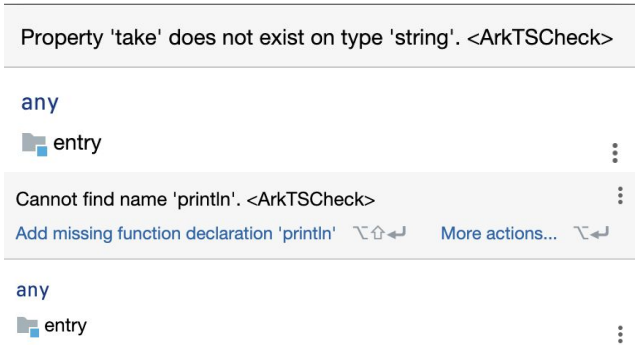
Трансляция вызовов библиотек

Kotlin

```
fun enterPin(
    pin: String,
) {
    val firstFourDigits = pin.take(4)
    println("PIN (first 4 digits): " + firstFourDigits)
    currentPin = pin
}
```

ArkTS

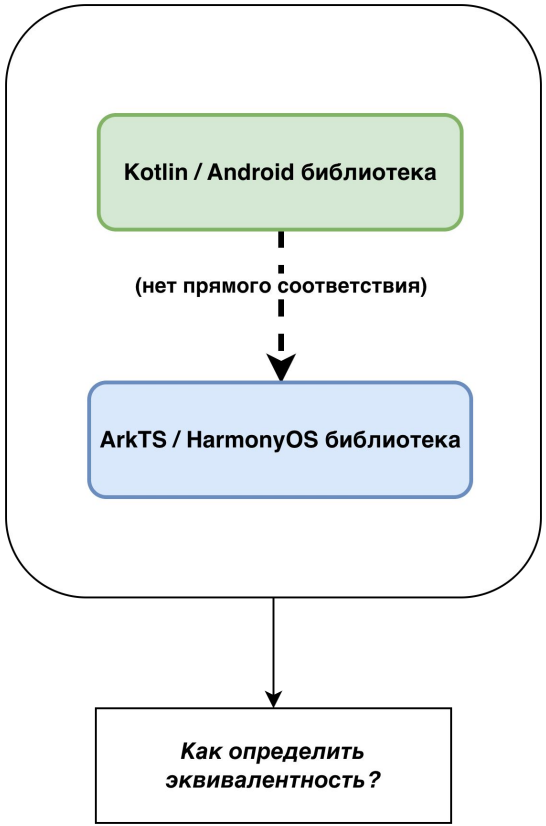
```
export function enterPin(pin: string): void {
    const firstFourDigits: string = pin.take(4)
    println('PIN (first 4 digits): ' + firstFourDigits)
    this.currentPin = pin
}
```



Трансляция вызовов библиотек

- В коде Android-приложений вызываются функции и классы из **стандартных библиотек Kotlin** и Android SDK, **которые не имеют прямых аналогов** в HarmonyOS NEXT.

- **Невозможно выполнить** простую **синтаксическую замену** вызовов — требуется определить соответствие функций двух платформ.



Формальные спецификации библиотек. LibSL

Спецификация видимого поведения функции является частичной спецификацией поведения функции и отражает основное предназначение функции, видимое извне.

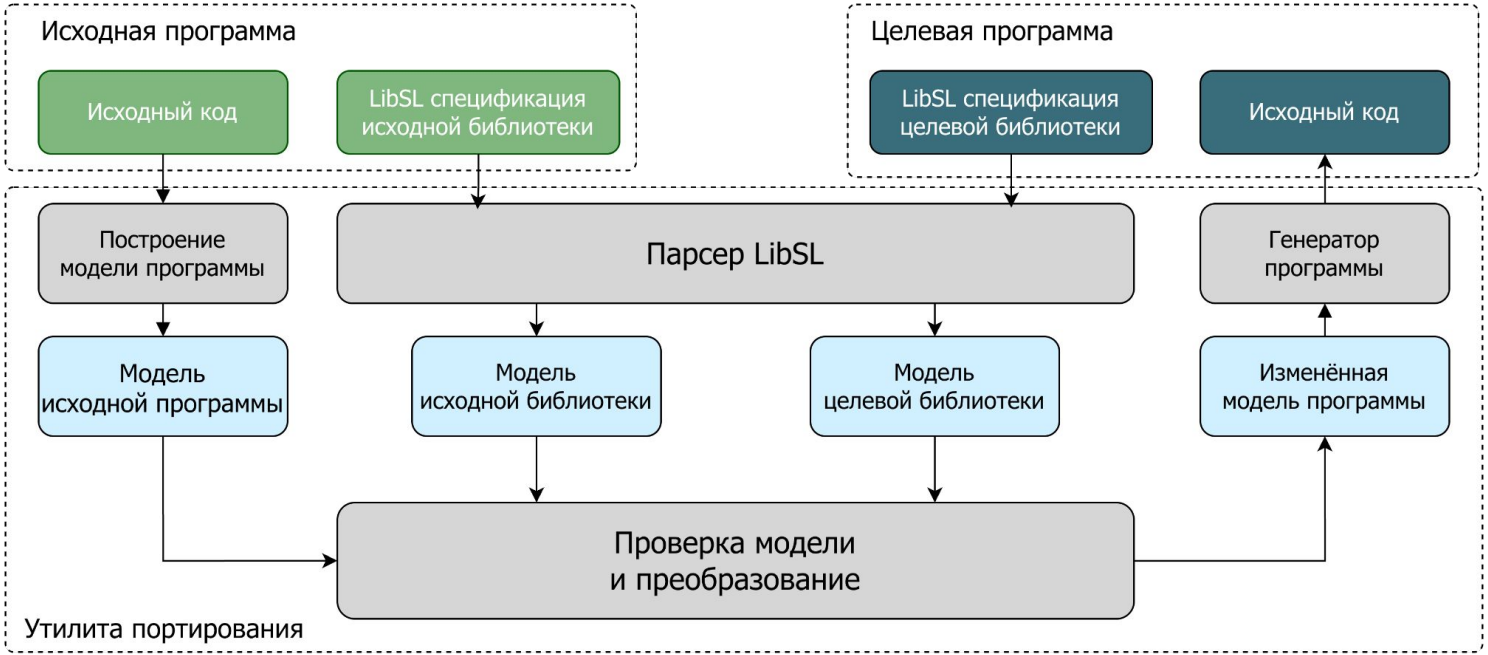
LibSL (Library Specification Language) — используется для декларативного **описания поведения** библиотек

```
libsl "1.0.0";
library `KotlinText`
    version "1.0.0";

typealias bool = bool;
typealias string = string;
typealias int = int32;
typealias double = float64;

define action STRING_TRIM(value: string): string;
define action STRING_TO_DOUBLE(value: string): double;

type kotlin.text { }

automaton kotlin.text: kotlin.text {

    fun isNotBlank(receiver: string): bool {
        var trimmed: string = action STRING_TRIM(receiver);
        result = trimmed != "";
    }

    fun toDoubleOrNull(receiver: string): double {
        result = action STRING_TO_DOUBLE(receiver);
    }
}
```

липтсофт

ИТМО ИНСТИТУТ ПРИКЛАДНЫХ КОМПЬЮТЕРНЫХ НАУК

Идея подхода



Идея подхода

Нужно привести **isNotBlank** к удобному формату и выразить через **String.trim**

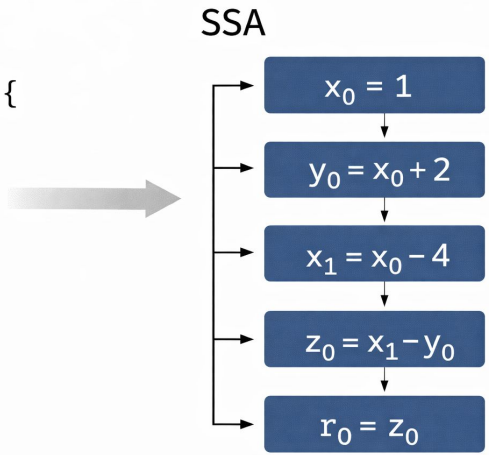
```
automaton kotlin.text: kotlin.text {

    fun isNotBlank(receiver: string): bool {
        var trimmed: string = action STRING_TRIM(receiver);
        result = trimmed != "";
    }
```

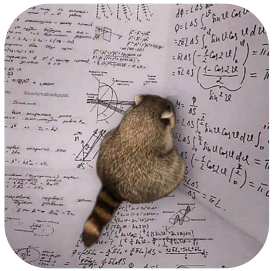
```
@Extension
automaton String.trim: String {

    fun call(receiver: string): string {
        result = action STRING_TRIM(receiver);
    }
}

@Extension
automaton String.toLowerCase: String {

    fun call(receiver: string): string {
        result = action STRING_TO_LOWER(receiver);
    }
}
```

SSA form

SSA form (Static Single Assignment form) — это представление программы, где **каждая переменная присваивается ровно один раз**.

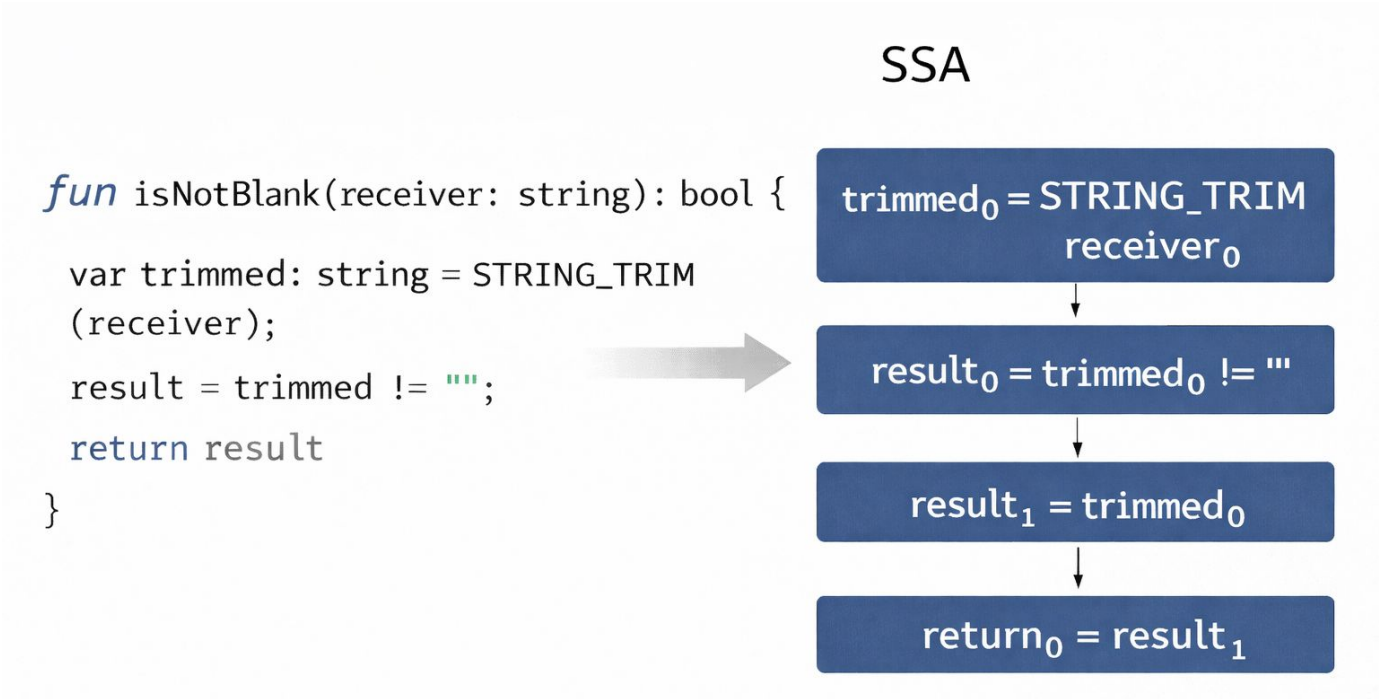
```
fun foo(): Int {
  x = 1
  y = x + 2
  x = x - 4
  z = x - y
  return z
}
```





SSA form


```
fun isNotBlank(receiver: string): bool {
  var trimmed: string = STRING_TRIM
  (receiver);
  result = trimmed != "";
  return result
}
```



Трасса исполнения

Трасса – *это упорядоченная последовательность шагов исполнения функции, зависящая от выполнения условных операторов.*

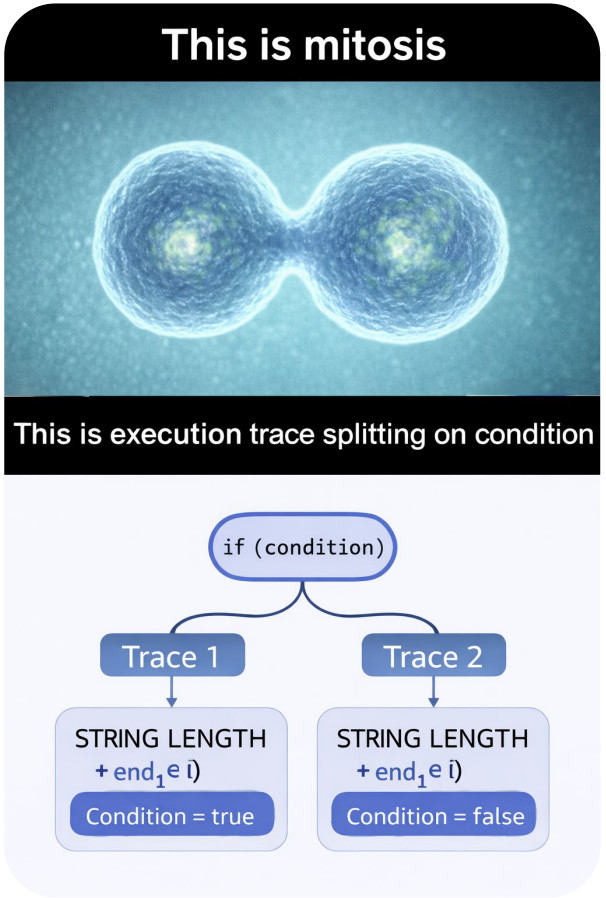
Последовательность Действий (**Action**), Условий (**Condition**) и Результатов (**Result**)

```
data class Trace(
    val function: LibFunction,
    val args: List<DeclStat>,
    val body: List<Step>
)

sealed class Step
data class Action(val step: ActionCallStat) : Step()
data class Result(val step: ResultStat) : Step()
data class Condition(val step: LExpr) : Step()
```

Структуры данных для описания трассы в коде

Трасса исполнения

Если есть условие (**if**) — трасса **раздваивается**. В одну трассу добавляется текущее её условие, в другую – его отрицание.



Пример получения трассы исполнения

Код

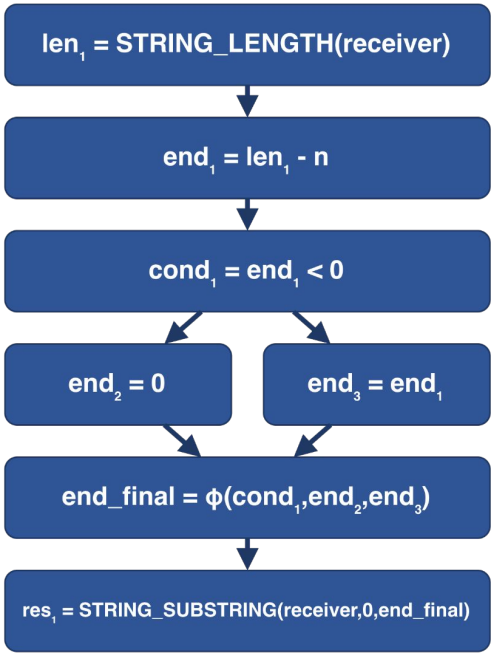
```
fun dropLast(receiver: string, n: int): string {
    length = STRING_LENGTH(receiver)
    end = length - n
    if (end < 0) end = 0
    result = STRING_SUBSTRING(receiver, 0, end)
}
```

Пример получения трассы исполнения

Код

```
fun dropLast(receiver: string, n: int): string {
    length = STRING_LENGTH(receiver)
    end = length - n
    if (end < 0) end = 0
    result = STRING_SUBSTRING(receiver, 0, end)
}
```

SSA

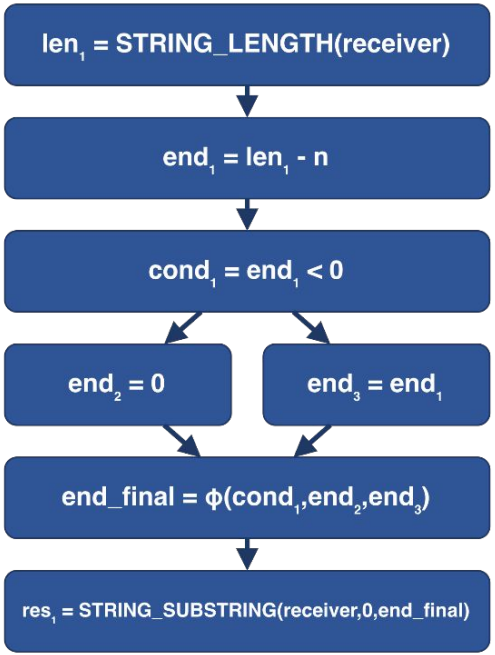


Пример получения трассы исполнения

Код

```
fun dropLast(receiver: string, n: int): string {
    length = STRING_LENGTH(receiver)
    end = length - n
    if (end < 0) end = 0
    result = STRING_SUBSTRING(receiver, 0, end)
}
```

SSA



Трассы

Trace 1

Action: STRING_LENGTH(receiver)
Condition: end1 < 0
Action: SUBSTRING(receiver,0,0)
Result: res1

Trace 2

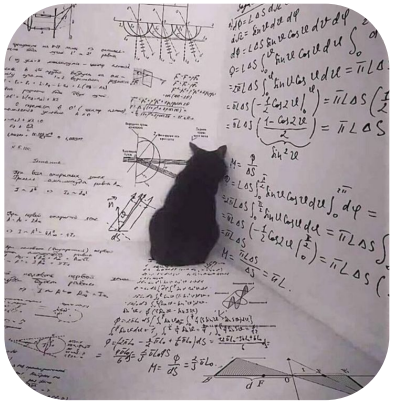
Action: STRING_LENGTH(receiver)
Condition: end1 ≥ 0
Action: SUBSTRING(receiver,0,end1-n)
Result: res1

SAT

SAT *(Boolean Satisfiability Problem) — задача булевой выполнимости: нужно определить, существует ли набор значений True/False для переменных, при котором логическая формула (в булевой логике) становится истинной.*

$$(x \lor y) \land (\neg x \lor z)$$

Есть ли такой набор значений **x**, **y** и **z**, при котором логическое выражение верно (**true**)



SAT

$$F = (x \vee y) \wedge (\neg x \vee z)$$

Итерация	x	y	z	(x ∨ y)	(¬x ∨ z)	F	Комментарий
0	?	?	?	?	?	?	Начальное состояние
1	**false**	?	?	?	**true**	**?**	Предположение: x = false
2	**false**	**false**	?	**true**	**true**	**false**	Предположение: y = false => **F = false**
3	**false**	**true**	?	**true**	**true**	**true**	Unit-propagation: y = true
4	**false**	**true**	**false**	**true**	**true**	**true**	Устанавливаем z, **SAT – решение найдено**

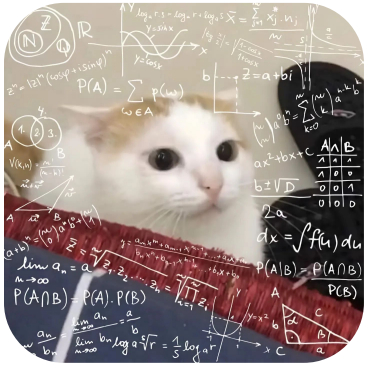
SAT -> SMT

SMT *(Satisfiability Modulo Theories) — проверка выполнимости логической формулы с учётом “теорий” (арифметика, массивы, строки, bit-vectors и т. д.).*

~~$(x \vee y) \wedge (\neg x \vee z)$~~

$$(x > 0) \wedge (x + y = 10) \wedge (y \geq 3)$$

Есть ли такой набор значений **x**, **y** и **z**, при котором логическое выражение верно (**true**)



SMT

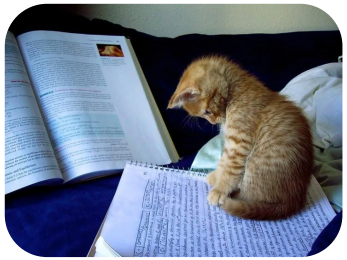
$$F = (x > 0) \land (x + y = 10) \land (y \geq 3)$$

№	x	y	x > 0	x + y = 10	y ≥ 3	F	Комментарий
0	?	?	?	?	?	?	Начальное состояние
1	-9999	?	**false**	?	?	**false**	Предположение: x = -9999 → **F = false**
2	9999	-9989	**true**	**true**	**false**	**false**	Предположение: x = 9999 → y = 10 − x = 10 - 9999 = -9989 → y ≥ 3 нарушено
3	?	3	?	?	**true**	?	Предположение: y = 3
4	7	3	**true**	**true**	**true**	**true**	x = 10 − y = 10 - 3 = 7 → **SAT — решение найдено**

KSMT

$$F = (x > 0) \land (x + y = 10) \land (y \geq 3)$$


```
(declare-fun x_0 () Int)
(declare-fun y_0 () Int)


(assert
  (and
    (> x_0 0)
    (= (+ x_0 y_0) 10)
    (>= y_0 3)
  )
)
```

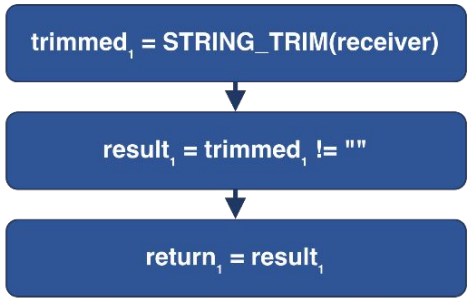


SSA -> Traces -> SMT Formula

Код

```
fun isNotBlank(receiver: string): bool {
    var trimmed: string = action STRING_TRIM(receiver);
    result = trimmed != "";
}
```

SSA



Трассы

Trace 1

Action: STRING_TRIM(receiver)
Result: $trimmed_1$!= ""

SSA -> Traces -> SMT Formula

Код

```
fun isNotBlank(receiver: string): bool {
    var trimmed: string = action STRING_TRIM(receiver);
    result = trimmed != "";
}
```

SSA

$trimmed_1$ = STRING_TRIM(receiver)

↓

$result_1$ = $trimmed_1$!= ""

↓

$return_1$ = $result_1$

Трассы

Trace 1

Action: STRING_TRIM(receiver)
Result: $trimmed_1$!= ""

$$(trimmed_1 = STRING\_TRIM(receiver)) \wedge (result_1 = (trimmed_1 \neq "")) \wedge (return_1 = result_1)$$

SSA -> Traces -> SMT Formula

Код	SSA	Трассы

```
@Extension
automaton String.trim: String {

    fun call(receiver: string): string {
        result = action STRING_TRIM(receiver);
    }
}
```

result$_1$ = STRING_TRIM(receiver)

return$_1$ = result$_1$

Trace 1

Action: STRING_TRIM(receiver)
Result: result$_1$

$$(\text{result}_0 = \text{STRING\_TRIM}(\text{receiver}_0)) \land (\text{return}_0 = \text{result}_0)$$

Код	SSA	Трассы

```
@Extension
automaton String.toLowerCase: String {

    fun call(receiver: string): string {
        result = action STRING_TO_LOWER(receiver);
    }
}
```

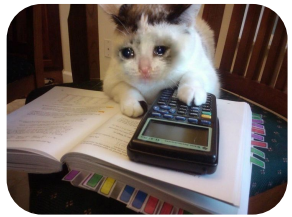
result$_1$ = STRING_TO_LOWER(receiver)

return$_1$ = result$_1$

Trace 1

Action: STRING_TO_LOWER(receiver)
Result: result$_1$

$$(\text{result}_0 = \text{STRING\_TO\_LOWER}(\text{receiver}_0)) \land (\text{return}_0 = \text{result}_0)$$



липтсофт

ИТМО ИНСТИТУТ ПРИКЛАДНЫХ КОМПЬЮТЕРНЫХ НАУК

Сопоставление SMT формул

Kotlin
`kotlin.text.isNotBlank`

$$(\text{trimmed}_0 = \text{STRING\_TRIM}(\text{receiver}_0)) \land (\text{result}_0 = (\text{trimmed}_0 \neq "")) \land (\text{return}_0 = \text{result}_0)$$

ArkTS
`String.trim`

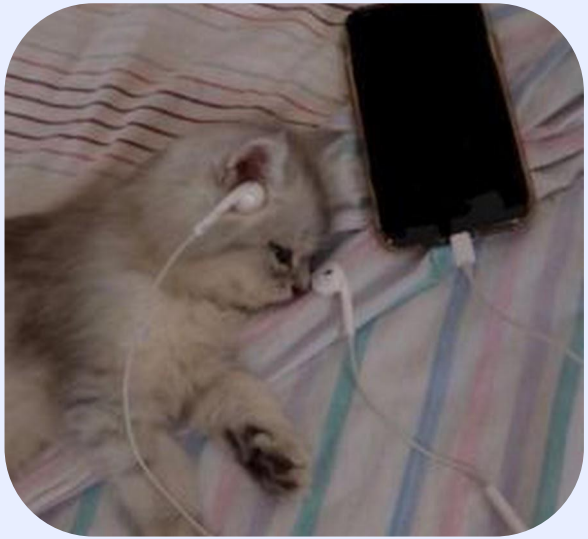
$$(\text{result}_0 = \text{STRING\_TRIM}(\text{receiver}_0)) \land (\text{return}_0 = \text{result}_0)$$

ArkTS
`String.toLowerCase`

$$(\text{result}_0 = \text{STRING\_TO\_LOWER}(\text{receiver}_0)) \land (\text{return}_0 = \text{result}_0)$$

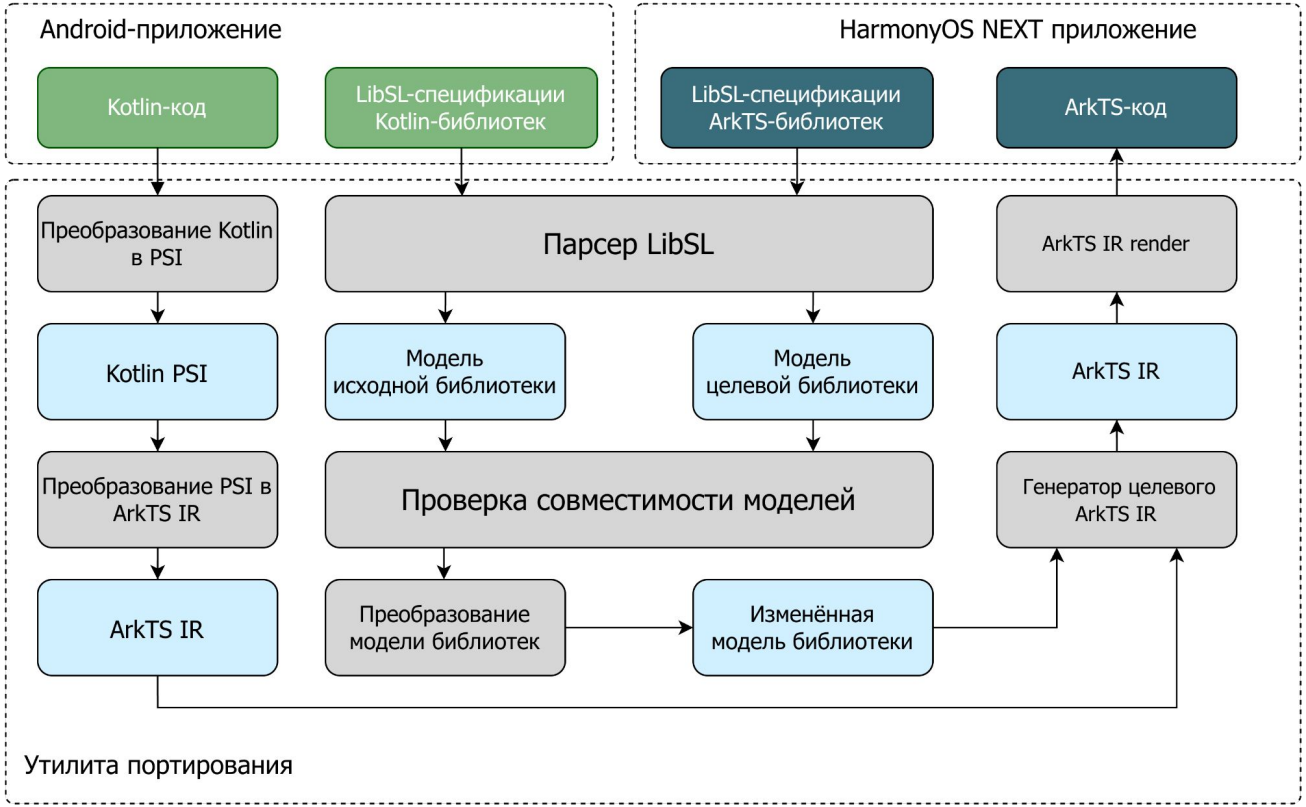
Анализ трасс через SMT

Остальное обсудим отдельно



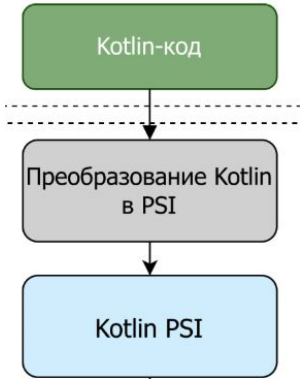
Я, пытающийся избавиться от математики в докладе...

Используемый подход



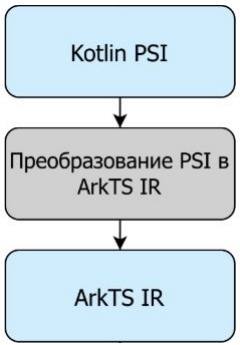
Используемый подход

```
fun enterPin(
    pin: String,
) {
    val firstFourDigits = pin.take(4)
    println("PIN (first 4 digits): " + firstFourDigits)
    currentPin = firstFourDigits
}
```




```
KtNamedFunction (name = "enterPin")
├── KtModifierList (empty)
├── KtTypeParameterList (null)
├── KtParameterList
|   └── KtParameter
|       ├── name = "pin"
|       └── typeReference
|           └── KtTypeReference
|               └── KtUserType
|                   └── Identifier("String")
├── KtTypeReference (null)   // return type inferred as Unit
├── KtBlockExpression
|   ├── KtProperty (val)
|   |   ├── name = "firstFourDigits"
|   |   ├── typeReference (null)
|   |   └── initializer
|   |       └── KtDotQualifiedExpression
|   |           ├── receiverExpression
|   |           |   └── KtNameReferenceExpression ("pin")
|   |           └── selectorExpression
|   |               └── KtCallExpression
|   |                   ├── calleeExpression
|   |                   |   └── KtNameReferenceExpression ("take")
|   |                   └── valueArgumentList
|   |                       └── KtValueArgument
|   |                           └── KtConstantExpression (4)
|   ├── KtCallExpression
|   |   ├── calleeExpression
|   |   |   └── KtNameReferenceExpression ("println")
|   |   └── valueArgumentList
|   |       └── KtValueArgument
|   |           └── KtBinaryExpression ("+")
|   |               ├── left
|   |               |   └── KtStringTemplateExpression
|   |               |       └── KtLiteralStringTemplateEntry
|   |               |           └── "PIN (first 4 digits): "
|   |               └── right
|   |                   └── KtNameReferenceExpression ("firstFourDigits")
|   └── KtBinaryExpression ("=")
|       ├── left
|       |   └── KtNameReferenceExpression ("currentPin")
|       └── right
|           └── KtNameReferenceExpression ("pin")
```

Используемый подход



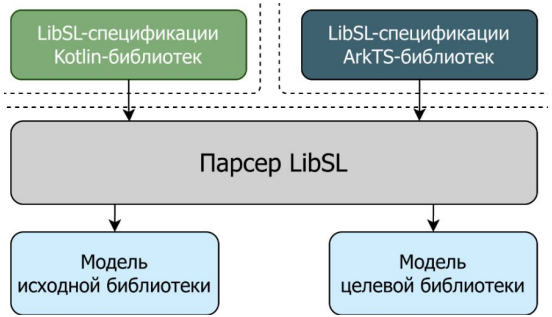

```
KtNamedFunction (name = "enterPin")
├── KtModifierList (empty)
├── KtTypeParameterList (null)
├── KtParameterList
|   └── KtParameter
|       ├── name = "pin"
|       └── typeReference
|           └── KtTypeReference
|               └── KtUserType
|                   └── Identifier("String")
├── KtTypeReference (null)   // return type inferred as Unit
├── KtBlockExpression
|   ├── KtProperty (val)
|   |   ├── name = "firstFourDigits"
|   |   ├── typeReference (null)
|   |   └── initializer
|   |       └── KtDotQualifiedExpression
|   |           ├── receiverExpression
|   |           |   └── KtNameReferenceExpression ("pin")
|   |           └── selectorExpression
|   |               └── KtCallExpression
|   |                   ├── calleeExpression
|   |                   |   └── KtNameReferenceExpression ("take")
|   |                   └── valueArgumentList
|   |                       └── KtValueArgument
|   |                           └── KtConstantExpression (4)
|   ├── KtCallExpression
|   |   ├── calleeExpression
|   |   |   └── KtNameReferenceExpression ("println")
|   |   └── valueArgumentList
|   |       └── KtValueArgument
|   |           └── KtBinaryExpression ("+")
|   |               ├── left
|   |               |   └── KtStringTemplateExpression
|   |               |       └── KtLiteralStringTemplateEntry
|   |               |           └── "PIN (first 4 digits): "
|   |               └── right
|   |                   └── KtNameReferenceExpression ("firstFourDigits")
|   └── KtBinaryExpression ("=")
|       ├── left
|       |   └── KtNameReferenceExpression ("currentPin")
|       └── right
|           └── KtNameReferenceExpression ("pin")
```

```
ArkTsTopLevelFunctionDeclaration (name = "enterPin")
├── visibilityModifier (null)
├── genericTypeParameters (null)
├── parameters
|   └── ArkTsParameterList.Parameter
|       ├── name = "pin"
|       └── type
|           └── DefaultArkTsType (rawType = "string", isNullable = false, generics = null)
├── returnType
|   └── DefaultArkTsType (rawType = "void", isNullable = false, generics = null)
└── body
    └── ArkTsExecutableBlock
        ├── ArkTsVariableDeclarationInstruction (const)
        |   ├── name = "firstFourDigits"
        |   ├── type = null
        |   └── initializer
        |       └── ArkTsDotQualifiedExpression
        |           ├── receiver
        |           |   └── ArkTsIdentifierExpression ("pin")
        |           └── selector
        |               └── ArkTsCallExpression
        |                   ├── callee
        |                   |   └── ArkTsIdentifierExpression ("take")
        |                   ├── genericTypeArguments (empty)
        |                   └── arguments
        |                       └── ArkTsLiteralExpression ("4")
        └── ArkTsExpressionInstruction
            └── GeneratedArkTsCallExpression
                ├── callee
                |   └── ArkTsIdentifierExpression ("generated_println")
                ├── genericTypeArguments (empty)
                └── arguments
                    └── ArkTsBinaryExpression (operator = ADD)
                        ├── left
                        |   └── ArkTsLiteralExpression ("\"PIN (first 4 digits): \"")
                        └── right
                            └── ArkTsIdentifierExpression ("firstFourDigits")
```

липтсофт

ИТМО ИНСТИТУТ ПРИКЛАДНЫХ КОМПЬЮТЕРНЫХ НАУК

Используемый подход




```
automaton kotlin.text: kotlin.text {

    fun take(receiver: string, n: int): string {
        result = action STRING_SUBSTRING(receiver, 0, n);
    }
}
```

```
@Extension
automaton String.substring: String {

    fun call(receiver: string, start: int, end: int): string {
        result = action STRING_SUBSTRING(receiver, start, end);
    }
}
```

липтсофт

ИТМО ИНСТИТУТ ПРИКЛАДНЫХ КОМПЬЮТЕРНЫХ НАУК

Используемый подход

Код

```
automaton kotlin.text: kotlin.text {

    fun take(receiver: string, n: int): string {
        result = action STRING_SUBSTRING(receiver, 0, n);
    }

}
```

Трассы

Trace 1

Action: STRING_SUBSTRING(receiver, 0, end)
Result: $result_1$

Код

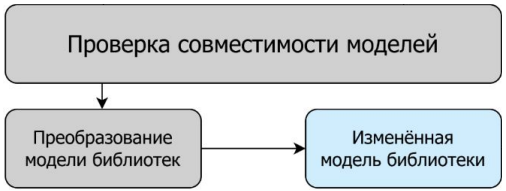
```
@Extension
automaton String.substring: String {

    fun call(receiver: string, start: int, end: int): string {
        result = action STRING_SUBSTRING(receiver, start, end);
    }
}
```

Трассы

Trace 1

Action: STRING_SUBSTRING(receiver, start, end)
Result: $result_1$

SAT!



** Аналогично и для println(...)/console.log(...)*

липтсофт

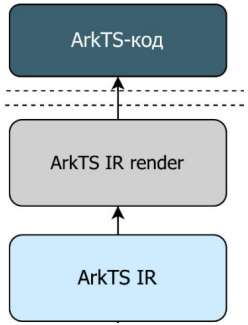
ИТМО
ИНСТИТУТ ПРИКЛАДНЫХ КОМПЬЮТЕРНЫХ НАУК

Используемый подход

```
take(n) => substring(0, n)
```

```
ArkTsDotQualifiedExpression
├── receiver
│   └── ArkTsIdentifierExpression ("pin")
└── selector
    └── ArkTsCallExpression
        ├── callee
        │   └── ArkTsIdentifierExpression ("substring")
        ├── genericTypeArguments (empty)
        └── arguments
            └── ArkTsLiteralExpression ("0")
            └── ArkTsLiteralExpression ("4")
```

Используемый подход




```
ArkTsTopLevelFunctionDeclaration (name = "enterPin")
├─ visibilityModifier (null)
├─ genericTypeParameters (null)
├─ parameters
│  └─ ArkTsParameterList.Parameter
│     ├─ name = "pin"
│     └─ type
│        └─ DefaultArkTsType (rawType = "string", isNullable = false, generics = null)
├─ returnType
│  └─ DefaultArkTsType (rawType = "void", isNullable = false, generics = null)
└─ body
   └─ ArkTsExecutableBlock
      ├─ ArkTsVariableDeclarationInstruction (const)
      │  ├─ name = "firstFourDigits"
      │  ├─ type = null
      │  └─ initializer
      │     └─ ArkTsDotQualifiedExpression
      │        ├─ receiver
      │        │  └─ ArkTsIdentifierExpression ("pin")
      │        └─ selector
      │           └─ ArkTsCallExpression
      │              ├─ callee
      │              │  └─ ArkTsIdentifierExpression ("substring")
      │              ├─ genericTypeArguments (empty)
      │              └─ arguments
      │                 ├─ ArkTsLiteralExpression ("0")
      │                 └─ ArkTsLiteralExpression ("4")
      └─ ArkTsExpressionInstruction
         └─ GeneratedArkTsCallExpression
            ├─ callee
            │  └─ ArkTsIdentifierExpression ("generated_println")
            ├─ genericTypeArguments (empty)
            └─ arguments
               └─ ArkTsBinaryExpression (operator = ADD)
                  ├─ left
                  │  └─ ArkTsLiteralExpression ("\"PIN (first 4 digits): \"")
                  └─ right
                     └─ ArkTsIdentifierExpression ("firstFourDigits")
```

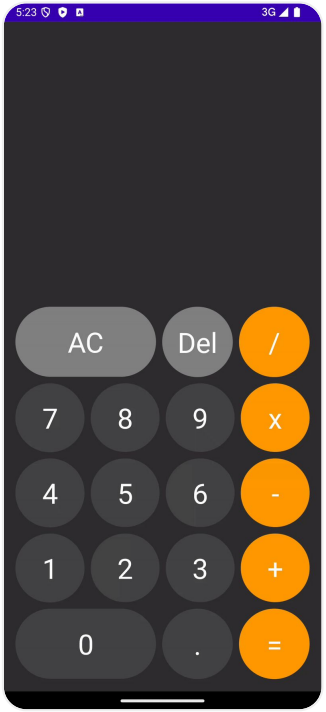
```
export function enterPin(pin: string): void {
    const firstFourDigits = pin.substring(0, 4);
    generated_println("PIN (first 4 digits): " + firstFourDigits)
    this.currentPin = firstFourDigits
}

function generated_println(value: string): void {
    console.log(value)
}
```

липтсофт

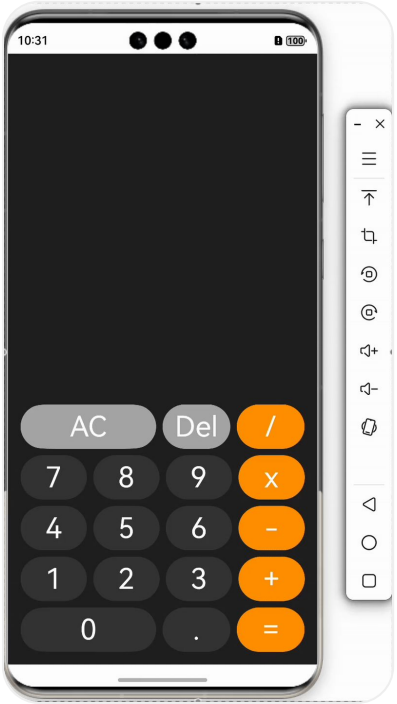
ИТМО
ИНСТИТУТ ПРИКЛАДНЫХ КОМПЬЮТЕРНЫХ НАУК

Оглавление

Портирование простого приложения



Android



HarmonyOS NEXT

Портирование простого приложения

	Android	HarmonyOS NEXT
Количество классов	**6**	**14**
Объем бизнес-логики	**120 строк**	**270 строк**
Количество методов, описанных через формальные спецификации	**6**	**6**
Время автоматического портирования	**2,5 с**	

github.com/<ТУТ_БУДЕТ_ССЫЛКА>

Итоги

Что сделали

- Портирование базовых конструкций **Kotlin** в **ArkTS**
- Портирование множества компонентов **Jetpack Compose** в **ArkUI**
- Портирование вызовов библиотек при помощи формальных спецификаций

Что планируем сделать

- Портирование навигации
- Портирование архитектуры проекта (многомодульность, директории и прочее)
- Портирование системных вызовов

липтсофт

ИТМО
ИНСТИТУТ ПРИКЛАДНЫХ КОМПЬЮТЕРНЫХ НАУК

Время вопросов


@MILKAsuper


svakun@gmail.com


github.com/MILKA-TOP